

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2007-513402

(P2007-513402A)

(43) 公表日 平成19年5月24日(2007.5.24)

(51) Int. Cl.	F I	テーマコード (参考)
G06F 21/24 (2006.01)	G06F 12/14 520B	5B017
H04Q 7/38 (2006.01)	H04B 7/26 109R	5K027
H04M 11/00 (2006.01)	H04M 11/00 302	5K067
H04M 1/67 (2006.01)	H04M 1/67	5K201
	G06F 12/14 520A	
審査請求 未請求 予備審査請求 未請求 (全 52 頁)		

(21) 出願番号 特願2006-537443 (P2006-537443)
 (86) (22) 出願日 平成16年11月8日 (2004. 11. 8)
 (85) 翻訳文提出日 平成18年6月21日 (2006. 6. 21)
 (86) 国際出願番号 PCT/GB2004/004701
 (87) 国際公開番号 W02005/046272
 (87) 国際公開日 平成17年5月19日 (2005. 5. 19)
 (31) 優先権主張番号 0325883.7
 (32) 優先日 平成15年11月6日 (2003. 11. 6)
 (33) 優先権主張国 英国 (GB)
 (31) 優先権主張番号 0329520.1
 (32) 優先日 平成15年12月19日 (2003. 12. 19)
 (33) 優先権主張国 英国 (GB)

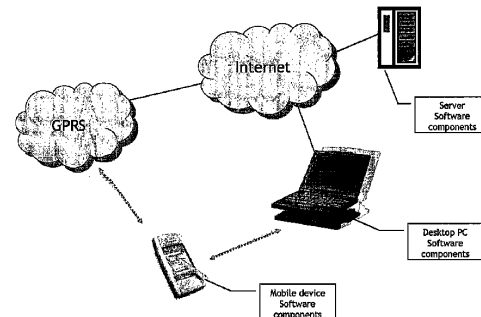
(71) 出願人 504070309
 インテュウェーブ リミテッド
 INTUWAVE LIMITED
 イギリス国 パークシャー エスエル6
 1 エヌジェイ, メイデンヘッド, ザ
 ブロードウェイ, シエナ コート (番地
 なし)
 Siena Court, The Br
 oadway, Maidenhead,
 Berkshire SL6 1NJ,
 United Kingdom
 (74) 代理人 100076428
 弁理士 大塚 康德
 (74) 代理人 100112508
 弁理士 高柳 司郎

最終頁に続く

(54) 【発明の名称】 携帯電話機上のリソースへのセキュア・マルチエンティティ・アクセス

(57) 【要約】

携帯電話機上の特定リソースへのアクセスの制御で、以下のステップから構成されることを特徴とする。(a) アイデンティティと許可状態を関係付けるが、ここでアイデンティティはリソースを使える可能性を持つ幾つかあるエンティティの1つに適用できる識別名であり、許可状態はリソースを実際に使用可能とするか否かを決定する。(b) リソースの使用を許可する許可状態に関係付けられたアイデンティティを持つ1つまたは複数のエンティティだけに、リソースの使用を許可する。



【特許請求の範囲】**【請求項 1】**

携帯電話機上の特定のリソースへのアクセスを制御する方法であって、

(a) 前記リソースを使える可能性を持つ幾つかのエンティティの1つに適用できる識別名であるアイデンティティと、前記リソースを実際に使用可能な許可状態とするか否かを決定し、前記アイデンティティと前記許可状態とを関係付けるステップと、

(b) 前記リソースの使用を許可する許可状態に関係付けられたアイデンティティを持つ1つまたは複数のエンティティだけに、前記リソースの使用を許可するステップと、
を備えることを特徴とする方法。

【請求項 2】

(a) 与えられたエンティティに関係付けられている、アイデンティティを持っているか、あるいは、アイデンティティを推定できる確実な署名を含んでいるスクリプト、または他の種類の実行可能なコードが特定のリソースの使用要求を送信するステップと、

(b) 前記携帯電話機上で実行されるソフトウェア・コンポーネントが、前記要求を処理し、前記スクリプトまたは実行可能なコードのアイデンティティに関係付けられている適用可能な許可状態を決定するためそのアイデンティティを使用するステップと、
を更に備えることを特徴とする請求項 1 に記載の方法。

【請求項 3】

前記許可状態が許可のタイプと値とを含むことを特徴とする請求項 1 または 2 に記載の方法。

【請求項 4】

与えられたアイデンティティに関係付けられている許可状態は更新または変更可能であることを特徴とする請求項 1 乃至 3 のいずれかに記載の方法。

【請求項 5】

許可状態の更新または変更が、前記携帯電話機から遠くにあるコンピュータから送信される命令により行われることを特徴とする請求項 4 に記載の方法。

【請求項 6】

前記リソースの使用には、アクセス、配備、変更、削除の1つまたは複数のものが含まれることを特徴とする請求項 1 乃至 5 のいずれかに記載の方法。

【請求項 7】

前記エンティティに関係付けられているスクリプトまたは他の種類の実行可能コードが、前記エンティティのアイデンティティとは別個または独立に追加されるアイデンティティを持ち、前記追加されるアイデンティティは前記スクリプトまたは前記コードを識別することを特徴とする請求項 2 に記載の方法。

【請求項 8】

前記アイデンティティが前記リソースの使用を許可されているかどうかにかかわらず、前記スクリプト自体が前記リソースの使用を許可されるかどうかの決定を可能にするため、前記追加されるアイデンティティに関係する許可状態が利用されることを特徴とする請求項 7 に記載の方法。

【請求項 9】

前記スクリプトまたは前記コードは前記アイデンティティを変更することが可能であることを特徴とする請求項 2 に記載の方法。

【請求項 10】

前記変更が遠隔のコンピュータから前記電話機に送信される命令の結果であることを特徴とする請求項 9 に記載の方法。

【請求項 11】

前記スクリプトまたは前記コードが所定の信頼水準で認証される場合だけ、前記アイデンティティはより高いまたはより広い許可状態に関係付けられているアイデンティティに変更されることを特徴とする請求項 9 または 10 に記載の方法。

【請求項 12】

前記方法は、オペレーティング・システムの一部でないコンポーネントにより前記携帯電話機に配備され、前記オペレーティング・システムを格納する前記電話機のメインROMに焼き付ける必要なしに前記電話機にインストール可能であることを特徴とする請求項2に記載の方法。

【請求項13】

前記コンポーネントは前記携帯電話機の安全なSIMの中で動作することを特徴とする請求項12に記載の方法。

【請求項14】

前記許可状態および異なるアイデンティティとの関係が前記SIMに保存され、前記コンポーネントは前記SIMの外で動作することを特徴とする請求項12に記載の方法。

10

【請求項15】

前記コンピュータから遠隔のコンピュータにより命令を送信することで、異なるアイデンティティに関係する許可状態を遠隔から管理するステップを更に備えることを特徴とする請求項14に記載の方法。

【請求項16】

前記コンポーネントが、異なるアイデンティティに関係する許可状態の一覧を、メモリに格納する、またはメモリから読み出すことを特徴とする請求項2に記載の方法。

【請求項17】

アイデンティティが、デジタル署名を使用する認証プロセスによりアクセスコードを捜すスクリプトのために決められることを特徴とする請求項2に記載の方法。

20

【請求項18】

前記認証プロセスがトークンとして転送可能なアイデンティティ・ハンドルを生成することを特徴とする請求項17に記載の方法。

【請求項19】

前記アイデンティティ・ハンドルが前記認証に基づき関連する信頼水準を有することを特徴とする請求項18に記載の方法。

【請求項20】

前記エンティティが個別のエンドユーザであることを特徴とする請求項1に記載の方法。

【請求項21】

前記エンティティがネットワーク・オペレータであることを特徴とする請求項1に記載の方法。

30

【請求項22】

前記エンティティが携帯電話機製造業者であることを特徴とする請求項1に記載の方法。

【請求項23】

前記エンティティがアプリケーション開発者またはベンダであることを特徴とする請求項1に記載の方法。

【請求項24】

前記エンティティが雇用主であることを特徴とする請求項1に記載の方法。

40

【請求項25】

前記エンティティが操作であることを特徴とする請求項1に記載の方法。

【請求項26】

開始コードが実行され、前記開始コードが特定のアイデンティティを持ち、前記アイデンティティに対する許可が開始時にできることとできないことを決定するため、前記操作が前記電話機を起動することを特徴とする請求項25に記載の方法。

【請求項27】

前記エンティティが開始タイマーの操作であることを特徴とする請求項1に記載の方法。

【請求項28】

50

前記エンティティがエンティティの種類またはタイプであることを特徴とする請求項 1 乃至 27 のいずれかに記載の方法。

【請求項 29】

少なくとも 2 つのエンティティが、互いに階層的に配列されている許可状態に関してアイデンティティを持たないことを特徴とする請求項 1 乃至 28 のいずれかに記載の方法。

【請求項 30】

前記エンティティが、互いに階層的に配列されている許可状態に関してアイデンティティを持たないことを特徴とする請求項 1 乃至 29 のいずれに記載の方法。

【請求項 31】

前記エンティティが前記電話機上の全リソースを使用する権利を自動的に持たないことを特徴とする請求項 1 乃至 30 のいずれかに記載の方法。

【請求項 32】

前記リソースが特定のデータであることを特徴とする請求項 1 乃至 31 のいずれかに記載の方法。

【請求項 33】

前記許可状態は、前記データの読み出し、修正、削除が可能かどうかを決定することを特徴とする請求項 32 に記載の方法。

【請求項 34】

前記リソースは特定の実行可能なアプリケーションであり、前記許可状態は前記アプリケーションが実行または更新可能か否かを決定することを特徴とする請求項 1 乃至 33 のいずれかに記載の方法。

【請求項 35】

前記リソースが前記電話機上のハードウェア・リソースであることを特徴とする請求項 1 乃至 34 のいずれかに記載の方法。

【請求項 36】

前記リソースが前記電話機上のネットワークまたは通信リソースであることを特徴とする請求項 1 乃至 35 のいずれかに記載の方法。

【請求項 37】

アイデンティティと許可状態を関係付けるステップは、前記電話機のメモリに格納される関係を記録することを特徴とする請求項 1 乃至 36 にいずれかに記載の方法。

【請求項 38】

前記リソースの使用を許可するステップは、前記電話機のデータを処理する CPU で実行されることを特徴とする請求項 1 乃至 37 のいずれかに記載の方法。

【請求項 39】

特定のリソースを持つ携帯電話機であって、前記リソースへのアクセスが請求項 1 乃至 38 のいずれかに記載の方法を用いることにより制御されることを特徴とする携帯電話機。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、スマートフォンならびに他の音声およびデータを扱えるモバイル機器等の携帯電話機上のリソースに安全なマルチエンティティ・アクセスを提供する方法に関するものである。

【背景技術】

【0002】

スマートフォンは新種のモバイル機器で、電話型の機器の中に移動可能な音声とデータの機能を組み込んでいる。そこには、新しいソフトウェア・アプリケーションをインストールし実行可能にするオペレーティング・システムと一緒に組み込まれている。現在普及しているスマートフォン・オペレーティング・システムはSymbian OS、Smartphone 2003

10

20

30

40

50

、PalmOSである。オペレーティング・システムは現在、シングルユーザ・オペレーティング・システムとして設計されており、シングルユーザの使用に最適化されている。その上、携帯電話業界は一般的に、リソースへのアクセスを明示する階層化された「オニオンスキン」モデルをモバイル機器のセキュリティに関する適切な図式と考えている。

内層：ユーザ、電話機所有者

次の層：オペレーティング・システム／ミドルウェア・ソフトウェアのベンダ

次の層：携帯電話機製造業者

外層：ネットワーク・オペレータ

この図式が意味することは、例えば、エンドユーザが、個人情報と（システムに接続するためのアクセスコードのような）銀行Aに属する情報を格納している銀行Aのバンキング・アプリケーションを実行していれば、エンドユーザはデータにアクセスできるが、そのモデルの内層以外の各層にいる誰でもデータにアクセスできる。ネットワーク・オペレータが望むなら、すべてのエンドユーザ情報にアクセスできる、ということである。この図式は次のことも意味する。ソフトウェア・ベンダのようなエンティティが、重要なシステムを展開しようとしている企業に本機器を売ろうとするなら、その企業は、携帯電話機製造業者とネットワーク・オペレータがアプリケーションとデータを結局のところ管理する図式を信じるか、アプリケーションを展開しないかのどちらかを強いられる。

【0003】

別の視点に立ち、携帯電話機をネットワークで結ばれたコンピュータと見なす。サイバー空間には数百万のユーザがいて、その電話機が実行しているアプリケーションに接続することが潜在的に可能である。「オニオンスキン」モデルでは、基礎知識が提供されていないし、オペレーティング・システムのサポートがないし、多数の着信接続がある状況でのアプリケーションの実行に関するアドバイスもない。電話機は、ユーザAはXを実行できるがユーザBはできないとどのように判断するか。簡単に言うと、アプリケーションは自分で決定しなければならない、各アプリケーションは独立しているのだから。アプリケーション・ベンダが問題の解決に失敗すると、サイバー空間にいるユーザBは、操作できるオニオンスキンが1層だけなので、理論的には電話機の所有者と同じレベルの特権でその電話機上のコードを実行できる。

【発明の開示】

【発明が解決しようとする課題】

【0004】

この2要因のもたらす結末は、(a)適切なセキュリティ・モデルがないので、組織はスマートフォンを採用しないか、(b)セキュリティ・サポートがないので、アプリケーションが開発されないか、(c)データの紛失や自分の電話機の違法使用に関するセキュリティ不安が知れわたった後、エンドユーザはスマートフォンの使用を恐れて使わなくなくなるか、あるいは3つのことが全部起こる。必要とされているものは、企業、アプリケーション開発者、エンドユーザ全部の要求を反映するセキュリティ・モデルである。

【課題を解決するための手段】

【0005】

最初の局面では、携帯電話機上の特定のリソースへのアクセスを制御する方法があり、以下のステップから構成されることを特徴とする。

(a)アイデンティティと許可状態を関係付けるが、ここでアイデンティティはリソースを使える可能性を持つ幾つかあるエンティティの1つに適用できる識別名であり、許可状態はリソースを実際に使用可能とするか否かを決定する。そして、
(b)リソースの使用を許可する許可状態に関係付けられたアイデンティティを持つ1つまたは複数のエンティティだけに、リソースの使用を許可する。

【0006】

アクセス管理への概念的に類似したアプローチは、ネットワーク化されたPCの世界では使用されているが、これまで誰もこのアプローチを携帯電話機に適用していない。その理由は、携帯電話はシングルユーザ機器という抗し難い思い込みがある。その結果、携帯電

10

20

30

40

50

話分野で働く熟練した実装者は、多数のエンティティが機器上のリソースにアクセスするのを可能にするアクセス管理技術の配備を通常考慮しようとししない。例えば、複数の異なるエンドユーザが、一人用のPCやサーバ上のデータやアプリケーションにアクセスできるようにしようとは考えない。

【0007】

本発明は、それゆえ、電話機の中で多数のエンティティが活動している状況を反映している。本発明は、「アイデンティティ」についての基本となる概念から始まる。「アイデンティティ」は個人、ユーザ、組織、1つのコード、サーバ等のエンティティに言及する方法である。エンティティのためにコードが機器上で実行される。異なるエンティティが1つのアイデンティティを共有できるが、異なるアイデンティティを持つこともできる。それゆえ、エンティティが多数活動中であることから、多数のアプリケーションを収容しているスマートフォンでは、多数のアイデンティティも活動することができる。電話機所有者、オペレータ、アプリケーション・ベンダ、所有者が働く企業のISマネージャー等の異なるエンティティに関係している異なるアイデンティティがある。従って、携帯電話機はシングルユーザ機器であると決めてかかる先行技術のアプローチと異なり、本発明は、多数のエンティティ、従って多数のアイデンティティが活動中であるとの着想を採用しており、各アイデンティティは何を実行することが許可されていて何を許可されていないかを示す一組の許可を持つように、電話機を設定できるようにしている。

10

【0008】

実装において、方法は以下のものから構成されることを特徴とする。

20

- (a) 任意のエンティティに関係付けられているスクリプトまたは他の種類の実行可能コードが、特定のリソースの使用要求を送信する。そのスクリプトはアイデンティティを持っているか、アイデンティティを推定できる確実な署名を含んでいる。
- (b) 機器上で実行されるソフトウェア・コンポーネントが、要求を処理し、そのスクリプトまたは実行可能コードのアイデンティティに関係付けられている適用可能な許可状態を決定するためそのアイデンティティを使用する。

【0009】

許可状態は、通常は許可のタイプとバリューを含む。任意のアイデンティティと関係付けられている許可状態は、例えば携帯電話機から離れたところにあるコンピュータから送信される命令により、更新または変更可能である。

30

【0010】

リソースの「使用」という用語には、アクセス、配備、変更、削除の1つまたは複数の操作を含む。

【0011】

任意のエンティティに関係付けられているスクリプトまたは他の種類の実行可能コードは、その任意のエンティティのアイデンティティとは関係関連のない、追加されるアイデンティティを持ってもよい。追加されるアイデンティティは、スクリプトまたはコードを識別する。UNIX（登録商標）アクセス管理のような従来のセキュリティ・システムは、許可とユーザリストを持っている。UNIX（登録商標）では、各プロセスは3つのアイデンティティを持っている。実行可能ファイルの所有者、それを実行している個人の使用しているID、一時的に他のIDに切り替えられているプロセスのために保存されているID。本発明の特徴で、スクリプト／コードが許可を求めるときは、2つのアイデンティティが提示される。それを実行している個人の実行IDおよびそのコードのID。このように、コード自体がその所有者または実行者とは関係のないアイデンティティを持っている。ソフトウェア・コンポーネントは、次にこの追加されたアイデンティティに関係付けられている許可状態を使用することができ、任意のエンティティがリソースの使用を許可されているか否かにかかわらず、スクリプト／コード自体がリソースの使用を許可されるかどうかを決定するために許可状態を使用できる。

40

【0012】

スクリプトまたはコードのアイデンティティは、遠くにあるコンピュータからその電話

50

機に命令を送信して変更することで、変更することができる。スクリプトまたはコードが所定の信頼水準で認証されている場合のみ、アイデンティティはより高いまたはより広い許可状態に関係付けられたアイデンティティに変更できる。

【0013】

本方法は、オペレーティング・システムの一部でないコンポーネントにより携帯電話機に配備可能であり、それゆえ、オペレーティング・システムを格納している電話機のメインROMに焼き付ける必要なしに電話機にインストールすることが可能である。このことは、既に出荷されている携帯電話機に本方法を配備することを可能にするので、極めて役に立つ。

【0014】

セキュリティを最適なものにするために、コンポーネントは携帯電話機の安全なSIMの中で実行できる。あるいはまた、許可状態並びに異なるアイデンティティと許可状態との関係付けは、SIMの外側で実行しているコンポーネントにより、SIMに格納できる。SIMカード・ハードウェア・トークンは、強力な暗号化を同時に行うと強力な認証レベルを提供するので、この方法は、単にソフトウェアを使用することでは達成できない、はるかに高いレベルのセキュリティを提供する。異なるアイデンティティに関係付けられている許可状態の遠隔管理は、そのコンピュータから遠くにあるコンピュータから命令を送信することにより可能である。

【0015】

コンポーネントは、異なるアイデンティティに関係付けられた許可状態のリストをメモリに蓄え、またメモリからそれを取り出す。さらに、アイデンティティは、デジタル署名を使用する認証プロセスによりアクセスコードを捜すどのスクリプトに対しても決定される。認証プロセスは、トークンとして転送可能なアイデンティティ・ハンドルを発生する。アイデンティティ・ハンドルは認証に関係する信頼水準を持っている。

【0016】

エンティティは、個別のエンドユーザ、ネットワーク・オペレータ、携帯電話機製造業者、アプリケーション開発者またはベンダ、雇用主、操作のいずれかである。エンティティが操作の場合は、特定のアイデンティティを持つ開始コードが実行され、そのアイデンティティのための許可が開始時に何を実行でき何を実行できないかを決定するため、操作が電話機を起動する。別の操作は、開始タイマーかもしれない。

【0017】

また、エンティティが、エンティティの種類やタイプのこともある。

【0018】

本発明のアプローチは、従来の「オニオンスキン」階層化モデルとは非常に異なる。それどころか、少なくとも2つのエンティティは、互いに関して階層的に配列された許可状態に関係付けられているアイデンティティを持たない。多くの例において、互いに関して階層的に配列されている許可状態に関係付けられているアイデンティティを持つエンティティはなく、電話機上のすべてのリソースを使用できる権利を自動的に持つエンティティもない。

【0019】

リソースは特定のデータのことがある。そのとき許可状態は、そのデータが読み出し、修正、削除が可能かどうかを決定する。また、リソースは特定の実行可能なアプリケーションのこともあり、そのとき許可状態は、アプリケーションを実行または更新可能かを決定する。リソースは、電話機のネットワークまたは通信リソースのような電話機上のハードウェア・リソースのこともある。

【0020】

ハードウェアとの物理的なやりとりに関しては、アイデンティティと許可状態を関係付けるステップは、関係の記録を電話機のメモリに蓄積して終わる。また、リソースの使用を許可するステップは、電話機の中のデータを処理するCPUで行われる。

【0021】

10

20

30

40

50

別の側面として、特定のリソースを持つ携帯電話機があり、その中ではリソースへのアクセスは上記の方法を使用し管理されている。

【0022】

(概念レベルの例)

いくつかの例が役に立つだろう。銀行口座の明細を収納している電話機上のファイルを考えてみよう。電話機の所有者は、そのデータにアクセスを試みるバンキング・アプリケーションを実行する。そこでは、所有者とバンキング・アプリケーションの2つのアイデンティティが活動中である。アクセス管理はソフトウェア・コンポーネントにより処理されるが、ソフトウェア・コンポーネントはカーネルであり、オペレーティング・システムに不可欠の部分ではない。これは、「mrixカーネル」である。カーネルは、所有者はデータに読み出しだけのアクセスが可能であり、さらにバンキング・アプリケーションもそのデータにアクセスが可能であると決定できる（これをユーザがダウンロードしたばかりのスペース・インベダー・ゲームと比較してみると、そのゲームは誰が実行するかにかかわらず、バンキング・データにアクセスできないことは間違いない）。

10

【0023】

バンキング・アプリケーションが本当に主張しているとおりのものであるか確認するために、デジタル署名を使用することができる。言い換えると、変装したインベダー・ゲームではない。さて銀行Aを考える。銀行Aは、展開したバンキング・アプリケーションを遠隔で管理したい。多分データを変更したい。電話機に接続し、バンキング・アプリケーション管理モジュールにログインする。「mrix」カーネルは、銀行が銀行口座の明細を所有しており、それゆえそのファイルに読み出し／書き込みアクセスができると決定する。

20

【0024】

一歩進めた例として、ユーザはセールスマンとして会社Bに出勤し、そのユーザの電話機に会社Bのセールス・アプリケーションを展開することを考えよう。今やその電話機上に複数のアプリケーション、複数のデータ源、複数の所有者が存在する。ユーザは、自分自身の個人的な連絡先、会社Bの連絡先、銀行Aの口座情報、会社Bの売上高を持っている。一組織が、そのすべての情報にアクセスできるべきではない。本発明は、誰もがアクセスすべき情報だけにアクセスすることを確実にする。

【0025】

もっと複雑な例をあげる。ユーザのボブがアリス銀行からバンキング・アプリケーションを手に入れる。そのバンキング・アプリケーションは彼の電話機に格納され、mネットワークを使用する。アプリケーションは、mrBankClientとmrBankAdminの2つのスクリプトから構成されている。ここで用語についての注意書き、本発明の実装には「パイプ・プロセッサ」を配備する。これは、C++またはスクリプト言語で書かれた小さな独立したモジュールで、様々なスマートフォンの機能をカプセルに包み込んでいる。機器に常駐のmrixパイプ・プロセッサには「m r」が前に付けられる。

30

【0026】

ボブはmrBankClientを実行して、彼の銀行口座の明細にアクセスし、収支を見て、支払いを済ます等することができる。アリス銀行は彼の口座を管理するためにmrBankAdminを遠隔から実行でき、例えば、新しいバンキング機能を可能にしたり、彼が銀行の顧客を辞めれば削除したりする。バンキング・アプリケーションがボブの電話機に提供されたとき、必要なアイデンティティと許可のすべての設定がカーネルのアイデンティティ・データベース（現在はidentity.iniと呼ばれるテキストファイル）に供給された。ここで、例を挙げて、そのシステムがどのように働くかを説明する。

40

【0027】

ボブがmrBankClientスクリプトを実行すると、簡潔なユーザ・インタフェース画面が現れる。デフォルトでは、ゲスト・アイデンティティで実行される。ボブがバンキング情報にアクセスを試みると、mrBankClientは、ゲスト・アイデンティティは銀行口座情報にアクセスすることが許可されているかカーネルに尋ねる。カーネルは調べ、NOの回答を返す

50

。

【0028】

ボブはユーザ・インタフェースでログオン方法を選ぶ。どのアイデンティティか述べる
ことと、それを裏付ける証拠を提示することを求められる。これは、ユーザ名とパスワ
ードの組み合わせでもよい。指紋のような生体情報（アイデンティティと証拠が一緒になっ
ている）でもよい。アイデンティティと証拠を単に決めてかかることもある。これはボブ
の電話機で、ボブは他の誰にも電話機を貸さないと銀行／ボブは信用していて、そして万
一盗まれてもロックされる場合である。mrBankClientスクリプトは、どのような方法でも
使い証拠を付けて、現在のアイデンティティを「BobIdentity」に設定するようカーネル
に求める。カーネルはアイデンティティ・データベースを調べ、許可を与えるか否かを決定
する。スクリプトを実行している現在のアイデンティティが、今やゲストからBobIdent
ityに切り替わっている。 10

【0029】

ボブは次に彼の明細を見ようとする。mrBankClientは、BobIdentityが明細を見ること
を許可されているかカーネルに尋ねる。回答はイエスで、ボブは彼の明細を見る。

【0030】

アリス銀行は、ボブに当座貸越を与えることを決める。これは、ボブの残高がゼロにな
ってもmrBankClientスクリプトが支払いできることを意味する。現在のところ、BobIdent
ityに対する「OverdraftLimit」許可はゼロに設定されている。銀行の端末を使い、アリス
銀行の従業員がボブの電話機上のmrBankAdminサービスに接続し、mrBankAdminスクリプ
トを実行する。銀行の従業員はmrBankAdminに証拠を提示する。多分またユーザ名とパス
ワードだ。そうして、現在mrBankAdminスクリプトがアイデンティティAliceBankIdentity
で実行されている。アリス銀行は「当座貸越追加」オプションを選ぶ。mrBankAdminスク
リプトは、今BobIdentityに対する許可を更新する必要がある。そこでmrBankAdminは、ユー
ザBobIdentityに対するOverdraftLimit許可を0から500に変更するようカーネルに依頼
する。これをするとき、現在のユーザのアイデンティティAliceBankIdentityの中とそれ
自体のアイデンティティmrBankAdminIdentityの中を通してゆく。mrBankAdminIdentityの
信憑性は、mrBankAdminスクリプトがmrBankAdminIdentityに一致する署名でサインされて
いる事実から保証されている。今やカーネルは、要求を許可するか否かを決めなければな
らない。最初にアイデンティティ・データベースを訪れ、「AliceBankAdminIdentityはBo
bIdentityの許可を変更する許可を持っているか」を尋ねる。二番目に、mrBankAdminIden
tityについて同じ質問をする。すなわち、それを誰が実行しているかにかかわらず、その
スクリプトはそれをするを許可されているか。必要な許可は前もって設定されている
のですべてはうまく行き、当座貸越は認められ、ボブは豪勢に散財できる。 20 30

【0031】

最後の例は、ベティがこっそりとボブの電話機にアクセスし、電話機が起動されたとき
にmrBankClientを実行するとウィルス・プログラムを実行するようにmrBankClientスクリ
プトを意地悪く変更する。ボブは気づかず、変更されたmrBankClientを正規のものと思い
実行する。ベティのコードが実行され、それはBOOTイベントに新しいコマンドを追加する
ためmrEventを使用することを要求する。この要求は、ボブがアプリケーションに正当に
ログインしているので、ボブのために実行される。しかし、すべてのものが失われはしな
い。mrBankClientスクリプトはmrEventを実行し、そこでmrEventはBobIdentityとスクリ
プト・アイデンティティが起動時にイベントを変更する許可を持っているか調べる。スク
リプトは変更され、それゆえ署名が照合できないので、スクリプト・アイデンティティは
mrBankClientIdentityとして提示されない。その代わりに、スクリプトはゲストとして提示
される。幸運にも、mrEventに対するブート・イベントの設定は、コマンドを誰が実施す
るかにかかわらず、実行コードの信憑性が確認された場合のみ成功することができるよう
に賢明に設定されている。すなわち、みだりに変更されていないバージョンのmrFile、ま
たはrshd（これはコマンド・インタプリタ・モジュールで、通常は起動時に開始するよう
設定されている）、またはユーザ・スクリプトは成功するが、みだりに変更されたバージ
 40 50

ョンが成功することはない。

【発明を実施するための最良の形態】

【0032】

A.1 mrixの概要

本発明は、インテュウェーブ社(Intuwave Limited)の「mrix」と呼ばれるソフトウェアを使い実装される。mrixは、モバイル機器用のネットワーク化されたアプリケーション・ソフトウェアの開発を迅速にできるようにし、そのような機器上のリソースへの安全なマルチエンティティ・アクセスを提供する。実装は、スマートフォン、デスクトップPC、サーバのようなモバイル機器を含むネットワークに接続している異なるコンピュータ機器に常駐するソフトウェアから構成される。構成例を図1に示す。

10

【0033】

ソフトウェア・コンポーネントは、迅速なアプリケーションの開発と配備を促進するため、ネットワーク上の異なる要素のすべてに必要とされる。このことを、モバイル機器上にネットワーク化されたアプリケーションを開発する次の例が説明している。アプリケーションは、ユーザがよりよい顧客関係を築くため、企業CRMシステムを十分に活用できるようにする。これをするため、ソフトウェアは、企業サーバに接続できるモバイル機器上で開発されなければならない。ここで企業サーバは、CRMシステムを実行し、企業と顧客のやりとりのすべてを管理する。モバイル機器は、サーバへ（GPRSのような）ワイドエリア接続およびPCのブロードバンド・ワイヤレス・リンク経由によるより高速なローカル接続の両方で接続できなければならない。モバイル機器のユーザ・インタフェースが限定されていることは、ユーザが自分のデスクにいるときはデスクトップPCの大きなスクリーンやキーボードを利用できるように、モバイル機器が容易にデスクトップPCに接続できなければならないことも意味する。

20

【0034】

そのようなアプリケーションを開発する従来の方法は、IDEのような適切な開発ツールを使いデスクトップPC上でソフトウェアを開発し、デスクトップPCのエミュレータ上でアプリケーションを実行しテストすることだろう。一旦ソフトウェアがエミュレータ上でうまく実行されると、次にモバイル機器に転送され、そこで再びデバッグが必要となる。このアプローチは、エミュレータとPC間に差がほとんどないので、ネットワーク化されていないアプリケーションには多くの場合適切である。しかし、ネットワーク化されるアプリケーションでは、モバイル機器が利用できる広範囲のネットワーク接続をエミュレータが持てないので、開発がはるかに難しい。この問題は、デスクトップPC（この用語には、Windows（登録商標）、Macintosh、Linux、他のオペレーティング・システムで動作するコンピュータを含んでいる）およびモバイル機器上にコンポーネントを持つことにより、mrixが解決する。ここでモバイル機器は、ブルートゥースのようなローカル・ワイヤレス・リンクにより局地的にまたはGPRS（またはSMSのような電話機への他の接続）を通じて遠隔から、どちらかのネットワーク接続上で実行可能である必要がある。これをもとにして、開発者はネットワーク化されるアプリケーションの開発を、以下のようにはるかに速いやり方で進めることができる。

30

1. 開発者は、そのアプリケーションにmrixパイプ・プロセッサ・コンポーネントのどのモジュール・セットを使用するか選ぶ。

40

2. 開発者は、選んだパイプ・プロセッサがコマンドラインからどのように使われるかテストする。

3. これらをまとめて、電話機上さらにはデスクトップPCから遠隔で実行できるすべての要素を含むアプリケーションにするため、簡単なスクリプトが編集される。

4. mrixの一部かもしれないmRouterのようなPC上の接続コンポーネントは、モバイル機器からデスクトップPCへネットワーク化された接続が必要とされたり、モバイル機器からデスクトップPCへのルーティングが必要とされたりすると使用される。mRouterの詳細についてはPCT//GB2002/003923参照のこと。その内容は参考文献に収録してある。

5. サーバ上の接続コンポーネントは、サーバから電話機に接続する必要があるなら使用

50

される。電話機のIPアドレスは外界からは見えず、サーバから連絡できないので、これが必要となる。それゆえ、サーバにネットワーク化された通信を可能にするため、中継サーバは電話機とバックオフィス・サーバの両方から見えることが必要とされる。

【0035】

mrixは、スマートフォンに関するソリューション作成における製品化期間を、下記により大幅に短縮するよう設計されたワイヤレス・ソフトウェア・プラットフォームである。

- ・習熟曲線を短縮し、それゆえ、より大きな開発者のコミュニティに開発を開放する。
- ・スマートフォンを共有のネットワーク・コンポーネントのように取り扱うことを可能にするネットワークOSのような設備を提供する。
- ・複雑なスマートフォンの機能をカプセル化する重要な「素材」を提供する。

10

【0036】

mrixは、スマートフォンのためのプラットフォームに寛容な遠隔コマンド実行環境を含んでいる。コマンド・インタプリタは、一組のコマンドまたは「パイプ・プロセッサ」を通じてスマートフォンと連結する。これらは、様々なスマートフォン機能をカプセル化している、C++またはスクリプト言語で書かれた小さな独立したモジュールである。機器常駐のmrixパイプ・プロセッサ（「mr」を前につけている）が提供され、それは多数のベアラ（GPRS、SMS、ブルートゥース、MMS、WiFi等）、（バーコード・リーダー、ペン、プリンタ、GPSのような）周辺機器、他の機器やサーバ、ネットワーク請求書作成の制御と管理を促進する。パイプ・プロセッサは、より多くの機能を作るため繋がる。これらの素材は、モバイル・ソリューションの迅速で繰り返しの多い開発を可能にする。スクリプト言語の使用は、はるかに広い範囲の開発者のコミュニティに開発を開放する。

20

【0037】

A2. mrix構成

mrixはスマートフォン上で動作するコマンドインタプリタおよび遠隔PCあるいは他の適切なプラットフォーム上で動作するコマンド実行シェルの周りに設計される。パイププロセッサを、mRouter（登録商標）を介してデスクトップPCから、あるいはリレーを介して遠隔サーバから（ユニックスコマンドのように）遠隔呼び出しができる。これによりmrixの解決法による開発およびデバッグを都合の良いデスクトップPCから実施できるばかりでなく、またネットワークを介して実行時にスマートフォンのコンポーネントの共有を許容する。

30

【0038】

幾つかのパイププロセッサは必須であり、システムの核と考えられる。実施例にはイベントに基づいて処理を開始および停止するのに使用されるmtEventあるいはmtATが含まれる。また、メモリ量を最小にするために、もし必要であれば実行時に削除できる選択可能なパイププロセッサの組も供給される。また、特注のパイププロセッサをC++あるいはLUAスクリプトで構築することができ、このためにテンプレートが提供される。

【0039】

A3. mrix解決法の実施例

使用するコンポーネントに関するより多くの情報については「mrixの特徴一覧」を参照されたい。

40

【0040】

【表 1】

予備部品の入手可能性の監視	
説明	現場の技術者が携帯する予備部品のレベルの正確な在庫管理の維持は困難である。ペンバーコードリーダーのような廉価なブルートゥース周辺機器をスマートフォンの高度な接続性の特徴と結合することにより、mr i xは現場サービス技術者がバンの在庫レベルの記録を取り、不足する在庫品目を地域の他のバンから取り寄せられるか自動的に問い合わせることを可能にする。
mr i x 解決法	mr B l u e t o o t hはスマートフォンとブルートゥースの駆動するバーコードペンとの間の接続の容易な管理に使用される。技術者が部品を必要とするとき、部品カタログの製品バーコードを「たたく」。持続すべき部品在庫はmr S t o r a g eを使用して装置で維持される。自動的にスマートフォンはバンで手に入る在庫を技術者に示す。もし部品が入手不可能であれば、mr M e s s a g eを介してSMSが創られ、他の技術者のスマートフォンに送信される。特定のSMSメッセージを受信して起動する受信者のスマートフォンのmr W a t c h F i l eを使用して、入力SMSは在庫チェックの実行を起動する。もし遠隔の技術者の電話機が、部品がバンにあることを示せば、SMSは送信元技術者に自動的に返信される。SMSを受信すると、スマートフォン上に自動的に指示(m r P r o m p t)が表示され、この指示は技術者に部品が入手できることを知らせ、部品を持つ技術者の電話番号を知らせる。mr S i mおよび現時点のセルIDを使用して処理をさらに高度化し、その地域の技術者から在庫入手の可能性をただ問い合わせるだけにする ことが出来る。
使用する コンポー ネント	R e l a y、mr B l u e t o o t h、mr S t o r a g e、mr M e s s a g e、mr W a t c h f i r e、M r P r o m p t、m r S i m

10

20

30

【 0 0 4 1 】

【表 2】

P CからSMSを送信	
説明	小型スマートフォンではテキストメッセージの入力は厄介である。装置のm r i xによりブルートゥースで接続されたP Cからテキストメッセージを作成し、電話機を介して送信することを可能にするアプリケーションを構築するのが簡明である。
m r i x 解決法	m-R o u t e rおよびm r C m dを使用してブルートゥースを介してスマートフォンをP Cに接続する。ユーザ認識（I d e n t i t i e s）の後、連絡先名と電話番号のリストが電話機から抽出され（m r C o n t a c t s）、P Cに表示される。ユーザはP C上の1または複数の連絡先を選択し、テキストメッセージの本文を入力し、「SMS送信」を押下する。P Cアプリケーションはデータを伴うm r M e s s a g eを呼び出し、テキストメッセージは電話機から自動的に送信される。
使用する コンポー ネント	m-R o u t e r、m r C m d、I d e n t i t i e s、m r C o n t a c t s、m r M e s s a g e

10

20

【 0 0 4 2 】

【表 3】

遠隔スマートフォンのサポート	
説明	遠隔のスマートフォンユーザへのサポートの提供は問題であることがあり、m r i xは遠隔P C（およびエンドユーザからの承認）を持つオペレータがセル方式ネットワークを介して接続されたスマートフォンを完全に制御することを許容する。
m r i x 解決法	エンドユーザはセル方式ネットワークを介してネットワークホストのR e l a yに自動的に接続するスマートフォン上でサポートアプリケーションを動作させる。また、オペレータはそのP C上のアプリケーションを介してR e l a yに接続する。一度全てが接続されると、オペレータは直接スマートフォンと接続することができる。m r K e y b o a r dおよびm r I m a g eを使用して、スマートフォンのスクリーンの実時間で動くイメージおよびスマートフォンのキーボードの動作する可視イメージがスマートフォンのスクリーンに表示される。m r P r o m p tを使用して、オペレータはユーザに装置上であるタスク実行の承認を求めることができる。m r P Sを使用して、オペレータはスマートフォントで、現時点で動作するアプリケーションリストを見ることができる。m r L a u n c hおよびm r S h u t d o w nを使用して、オペレータは処理動作の開始と停止および電話機を遠隔からリスタートさせることができる。m r S y s I n f oを使用して、オペレータは利用できるメモリ、記憶装置のタイプなどを含むスマートフォンに関する情報を見ることができる。全てのタスクは操作を通じて関係するユーザにより遠隔から完遂される。
使用する コンポー ネント	R e l a y、I d e n t i t i e s、m r K e y b o a r d、m r I m a g e、m r P r o m p t

10

20

30

【0043】

【表 4 - 1】

m r i x の特徴一覧		
所要事項	特徴	利点
PCの接続性	m-R o u t e r (登録商標) m r C m d	シリアルリンクを介する I P の提供。ブルートゥース、I r D A、ケーブルなどを介して接続された P C から装置の完全制御を許容。
遠隔接続 (G P R S、W i F i など) を介する操作	R e l a y	装置が「可視」でないファイアウォールで守られたネットワークを介する装置とサーバ処理の接続。G P R S ネットワークの装置が他の装置あるいはサービスにより見出されるのを許容し、「押す」を容易する。
セキュリティ	I d e n t i t i e s	ユーザ (あるいはサービス) は装置のコマンドにアクセスするために信用を供与しなければならない。
データ格納	S k y D r i v e (遠隔) m r S t o r a g e (ローカル)	スマートフォンに捉えられた重要データはネットワーク、あるいは装置の常時利用可能なバーチャル格納装置に送信できる。格納されたデータは後に処理できる。
メッセージ送信	m r M e s s a g e	装置のメッセージセンタの容易な監視と管理。
イベント駆動操作	m r W a t c h f i r e、 m r A t、 m r E v e n t	ある状況に遭遇すると動作を起動。例えば、特定の S M S / M M S メッセージを受信するとスクリーンを動作させる。また、特定時間に動作させる操作の計画。
接続性	m r B l u e t o o t h、 m r O b e x、m r T C P、 m r T h r o u g h p u t	多重ベアラを介する接続の創設と管理、送受信データの検査と処理およびネットワーク性能の測定。O B E X を介するファイルの送信。

10

20

30

40

【表 4 - 2】

電話機情報	mrSysInfo	利用可能な装置、空き空間、フォーマットなどを含む装置情報を返信。
ネットワーク情報	mrSIM	IMEI、現時点のセル-ID、地域および移動ネットワーク運用会社のようなネットワーク情報の返信。
遠隔制御	mrImage、 mrKeyboard	接続されたPCへの装置スクリーンの投影およびスマートフォンのキーボードの遠隔制御の許可。
ファイル操作	mrFile	スマートフォンファイルシステムの容易な操作。
通常の制御	mrPs、mrMr、 mrBoot、 mrShutdown、 mrLaunch	スマートフォンの処理の問い合わせ、開始および停止、ブートによるアプリケーションの自動開始および装置の閉鎖。
データ捕捉	mrPrompt	ポップアップダイアログボックスを介するスマートフォンユーザからのデータ捕捉
PIMデータアクセス	mrContacts、 mrAgenda	連絡、カレンダーおよびメモを含むスマートフォンPIMデータの完全検索、追加、編集および削除。vカード、ミニvカード、uid、スピードダイヤル、グループなどの操作可能。

【0045】

【表 5】

仕様		
サポートするオペレーティングシステム	スマートフォン	シンビアンOS v6. 1、7. 0、7. 0s、マイクロソフトポケットPCスマートフォンエディション
Relay、mrCmd		Mac OS X、リナックス、ウィンドウズ ME、2000、XPホームおよびプロフェッショナル

【0046】

A4. 特徴リスト

核をなすm r i xシステムは幾つかの要素を含み、その幾つかはスマートフォン上に配備される。

【0047】

m r c m d : m r c m d は2つの要素、スマートフォンのコマンドインタプリタおよび遠隔コマンド実行シェルからなる。コマンドインタプリタは現在シンビアン上で動作する。遠隔コマンド実行シェルはウィンドウズ、M a c O S Xおよびリナックス上で動作する。

m - R o u t e r (登録商標) : シンビアンOSのスマートフォン上のローカル接続の管理を処理するインタフェースの既存のm - R o u t e r (登録商標) 製品へのコマンドラインインタフェースである。m - R o u t e r (登録商標) はシリアル、ブルートウス、U S BおよびI r D Aベアラを介して動作する。

10

m r E l a y : m r E l a y はインタフェースの遠隔リレーサーバへのコマンドラインインタフェースとリレーサーバ自体の両方からなる。現在、リレーサーバはG P R Sあるいはローカルm - R o u t e r (登録商標) リンクにより代理されるW A Nを介してスマートフォンからアクセスすることができる。

パイププロセッサ : パイププロセッサはスマートフォンの機能をカプセル化する小型自己完結モジュールである。イベント処理およびファイルアクセスを管理する少数のパイププロセッサはm r i xの核に存在する。

スクリプトエンジン : 強力でコンパクト (60k) なL U A 5 . 0スクリプト記述エンジンはスマートフォンに含まれ、開発者がスクリプトを使用して直接難なくパイププロセッサの機能を結合することを許容する。スクリプト記述エンジンに含まれるのは、既存のパイププロセッサの機能を強力に結合する幾つかの核m r i xスクリプトである。

20

m r i x参照マニュアル : 全ての既存の核パイププロセッサの使用法を説明するH T M Lページである。m - R o u t e r (登録商標) およびm r c m d機能、並びに新しいパイププロセッサの記述に関する指示も存在する。文書およびスクリプトの詳細が含まれる。

【0048】

システムの核機能を拡張する付加的パイププロセッサの範囲がある。これらパイププロセッサは難なくm r i xシステムに付加され、その能力を高めることができる。

【0049】

A.5 適用分野

30

m r i x技術は、スマートフォン機器の遠隔制御が重要なところでは、広い範囲のアプリケーションに直接適用可能である。

【0050】

テスト : m r i xは、システム、機能、受け入れ、回帰、相互運用の各テストを完全自動化できる。

PIMアプリケーション : m r i xは、スクリプトがアクセスできるツールキットを提供することで、PC接続PIMアプリケーションの迅速な開発を可能にしている。

アクセス管理 : m r i xは、リソースへの安全なマルチエンティティ・アクセスを可能にする。

【0051】

40

A.6 利点

m r i xは、スマートフォン製造業者と電話ネットワーク・オペレータに非常に多くの利点を提供する。

開発スピード : m r i xの開発は、APIに対してコーディングするのではなく、スクリプトを素早く繰返し徐々に発展させることにより行われる。このことは、開発サイクルを大幅に高速化する。

コスト : m r i x機能はスクリプトに基づいているので、開発コスト、維持コスト、機能強化コストが大幅に減少している。

クロス・プラットフォーム : m r i xは、スマートフォンに対し完全なクロス・プラットフォーム・サポートを提供する。クロス・プラットフォーム・ツールキットと組み合わせる

50

と、サーバ・アプリケーションを異なるPCオペレーティング・システムにわたって実行できるように作ることができる。

【0052】

B. セキュリティ原則と哲学

mrrixのセキュリティは、実生活に合わせて作られている。実生活では、すべての人、すべてのものが、アイデンティティとして機能する。バンキングの例をもっと詳しく検討することは、mrrixの中でセキュリティが維持される方法を理解するうえで役立つ。名前とシナリオは露骨に作られているが、重要な点を説明するのに都合がよい。

【0053】

個人の「アリス」(A)が「ボブ国際銀行」(B)に入り、出納係「シャルロット」(C)の列に並ぶことを想定してみよう。それから、アリスは金を預けるか引き出す。これは単純な業務のようだが、個人の「イブ」(E)がアリスまたは銀行から金を盗もうとするかもしれない。イブが使うかもしれない方法を考えてみよう。アリスに扮して銀行から資金を引き出そうと企てるかもしれないし、偽の銀行でシャルロットの振りをするかもしれないし、実際に銀行に職を得てシャルロットとして銀行をだまして金を奪うかもしれない。銀行の外でイブがアリスを襲う荒々しい暴力による攻撃があるかもしれないし、銃を突きつけて銀行から金を奪うことも企てるかもしれない。

【0054】

アリス(A)がボブの電話機(B)上の連絡先にアクセスし、連絡先を追加または検索するためにmrContacts(C)を使うという、類似したシナリオを考えてみよう。「イブ」が連絡先を見たり改悪したりしようとすると考えよう。アリスに扮してボブの電話機を使う、またはアリスを連絡先に入れるため彼女に偽電話をするかもしれない。mrContactsとして働くが悪意を持って動作するソフトウェアを、ボブの電話機にインストールしようとさえするかもしれない。アリスまたはボブの電話機の物理的なハードウェアに対し、荒々しい暴力攻撃もあるかもしれない。

【0055】

冷淡に主張すると、両方のケースにおいてアリスの身体の保護は彼女自身の問題であり、銀行／電話機の保護の範囲外である。同様に、銀行建物／金庫および電話機の物理的メモリの物理的保護は、建物／チップの設計者の領域である。すべてのセキュリティやすべてのものごとのように、完全な答えはなく、絶対的なセキュリティもない。あるのは単なる信頼の程度である。

【0056】

このことは、私達が取組むことができる両例に共通のいくつかの重要なポイントを指摘する。

- ・アリスが彼女の主張しているとおりの人間であると証明できるようにしたい。
- ・出納係／mrContactsが信頼できることを保証したい。
- ・銀行／電話機が意図したものであり、贋物でないことを保証したい。
- ・アリスと出納係／mrContactsとのやりとりは盗聴されず、後で不正なアクセスに使用されることはないことを保証したい。
- ・ややこしいと銀行／電話機は使われないので簡単であることを保証したい。

【0057】

もう1つの例がある。それは銀行／電話機の顧客でない人物が何かしたい、多分硬貨を替えたいまたは公開の連絡先をチェックしたいというものである。このケースでは明らかに、ユーザを認証する必要はない。以前にかかわりがないので、認証できない。私達は、ゲストが何かすることを許可でき、他のことは禁止できるようにしたい。

【0058】

金が預けられたり移動されたりしたときに表示し、誰がその業務を認可／実施したかをチェックするため、銀行にとって監査は重要である。同様に電話機では、請求書作成や診断のため、あるいは単なる好奇心から、どの操作が、いつ誰により行われるかを記録したいという要望があるかもしれない。

10

20

30

40

50

【0059】

アリスが彼女が誰であることを証明することをちょっと考えてみよう。彼女に暗証番号または秘密のパスワードだけを提供し、銀行または電話機を使うときに毎回これを提示するように頼む。この方法は、イブがパスワードを推測したり立ち聞きしたりしないことを当てにしている。オンラインバンキングでよく使われるソリューションは、長い暗証番号を与え、数字のサブセットだけを尋ねることである。異なるサブセットを毎回尋ねることにより、秘密全体を発見するためにイブはより多くのやりとりを盗聴しなければならない。別の方法として、アリスの署名を使うこともできる。でも、イブがその署名のフォト・コピーを作ったらどうなるか。その心配については、出納係の前でアリスにサインして貰い、次に信頼できるコピーと比較するほうが安全である。公開鍵暗号技術を用いて、アリスは秘密鍵を使い任意のデータを暗号化し、彼女の公開鍵と一緒に暗号化したデータを電話機に与えることが可能である。公開鍵だけがデータを解読できるので、暗号者はアリスだと電話機は合理的に決めてかかれる。暗号化するために一回限りのデータを発生することにより、イブが彼女自身の邪悪な計画のために署名されたデータを捕まえても再利用することができないと、電話機は保証することができる。もちろん、この保証のすべては、アリスが相当する秘密鍵を持っているからである。銀行は、アリスが毎回同じ署名を使うなら、彼女が訪問を繰返すと、アリスを認識する。現実の世界では、この署名がある個人に一致すると証明するためには、銀行は運転免許証やパスポートのようなアイデンティティを証明するものを必要とする。同様に、デジタル署名は、ユーザが「前回」と同じユーザであることを電話機がわかるようにする。この署名を現実の世界のアイデンティティと関係付けるために、署名は別の信頼できるエンティティからの証明書により裏付けられる必要がある。そのような仕組みは、少なくとも信頼できる根本をなすエンティティに頼らざるを得ない。

【0060】

アリスは、出納係が信頼できるとしたらわかるだろうか。第一に、従業員を雇いリソースにアクセスできるようにするとき銀行は注意深い、と信頼しなければならない。出納係のシャルロットは彼女が主張するとおりの人であると、銀行が毎朝確認すると信頼しなければならない。たいていの銀行システムは、シャルロットにアリスの秘密のすべてを見させない。シャルロットは、アリスに尋ねる質問を表示するかもしれない端末の前に座る。シャルロットは回答を入れるかもしれない。しかし、通常シャルロットはすべての秘密を見ることはなく、見ることができるのはアリスを認証するために使われるサブセットだけである。このことは、シャルロットは認証の専門家である必要はなく、バンキング・システムに代わって質問し、アリスの回答を入力する必要があるだけである。mrixの地に戻ると、人は複雑な暗号文や各モジュールの中の認証コードを再生産したいとは思っていない。それどころか、rshdまたはmrBluetoothのようなモジュールは、信頼されているセキュリティ・モジュールに代わり質問をすることができ、アリスの回答をこの安全なモジュールに返し、成功すればアイデンティティとしてアブストラクト・トークンを獲得する。銀行は全顧客についての許可と情報のすべてを、すべての出納係が知ることを望まない。通常は認証後、出納係がその顧客情報のサブセットを見ることを可能にする。出納係は、情報の断片のすべてにはアクセスできず、手元のタスクに関係するものだけにアクセスできる。同様に、出納係のシャルロットは任意のユーザの情報にアクセスできず、認証されているユーザの情報だけである。アリスを認証する異なる方法があるかもしれないが、このステップは明らかに独立しているがアリスのアイデンティティを実際に利用して連続して起こる。mrixでは、各コンポーネントはユーザにチャレンジしアブストラクト・アイデンティティ・モジュールを入手するため、コア・セキュリティ・モジュールを使用できる。そのモジュールを使って、mrContactsまたはどのモジュールもアイデンティティ・ポリシーと許可についてシステムに質問できる。最後に、システムはモジュールとそれを「信頼する」というコマンドだけをロードする。

【0061】

C. ターゲット・システム

本セクションは、mrixの目標とするセキュリティを記述する。記載しているものがまだ存在していないかもしれないが、現在形で書く。別の文書は現在の状況を記述する。

【0062】

C.1 基礎となる実装

mrixの中核は以下に基づいている。

- ・mSecure — このシングルEPOCサーバはmStream、mIdentity、mrBootの機能を結合して一体化し、そうする中でオーバーヘッドが少なく、より強固なセキュリティとより多くの機能を提供する。例えば、パイプ・プロセッサを要求するのに、以前は2つの異なるサーバに少なくとも12回の呼び出しが必要だったが、今はたった2回だけだ。

【0063】

各種のラッパー／ヘルパーのライブラリがこれらコア・システムの上に層をなしている。

【0064】

C.2 アイデンティティ

mrixのセキュリティは、アイデンティティの概念に基づいている。アイデンティティは、ユーザIDや原則と類似して抽象概念である。パイプ・プロセッサはアイデンティティのために呼び出され、それゆえにそれがどのアイデンティティかにより異なる振る舞いを行うことができる。アイデンティティは、望むならモジュールとプロセス間で手渡されるアイデンティティ・ハンドル（トークン）によりシステムの中で代理される。アイデンティティ・ハンドルは、認証プロセスの結果としてだけ作成される。アイデンティティが与えられると、モジュールはそのユーザに対する許可と他の設定を尋ねることができる。モジュールを呼び出すアイデンティティに加えて、モジュールが利用できる2番目のアイデンティティもあり、それはそのモジュールを作成した／書いた／署名したアイデンティティである。

【0065】

C.3 パイプ・プロセッサ・メタデータ

各パイプ・プロセッサ（スクリプトを含むがそれに限定されない）は独立したメタデータ・ファイルと一緒に供給されることがある。このファイル自体が署名されているので、メタデータへの変更は見破ることができる。メタデータは以下のものを含む。

- ・ファイルが通信するパイプ・プロセッサのハッシュ
- ・著作権、バージョン、ライセンス、著者連絡先情報
- ・パイプ・プロセッサが必要とする許可とモジュール（換言すると依存関係）
- ・パイプ・プロセッサに埋め込まれている許可（換言するとポリシー）

ファイルの内容は、当初は著者によりテキスト文書として作成され、次にtxt2mrixユーティリティにより署名／編集される。

【0066】

C.4 認証

mrixは、秘密／公開鍵認証、プレーンテキスト認証、ハッシュ認証をはじめとする数タイプの認証を提供する。そのような証明物は、機器のメモリ上に暗号化されて格納されている。暗号化は、信頼できるハードウェア（つまりSIMカード）と組み合わせて、それとの連携でさらに強化される。

ユーザを認証するために、rshdはmrCmdを実行している遠隔のピアにチャレンジする。認証方式はサーバとピアの間にネゴシエーションされる。

認証メカニズムは以下のものを含む。

- ・プレーンテキスト・ユーザ名／パスワード—簡単だが、秘密もはっきり見えるかもしれない。

- ・ユーザ名／パスワードのMD5ハッシュとチャレンジ認証が必要とされるときは毎回唯一のキーがチャレンジに用いられる。アルゴリズムはこのキーおよび信頼されているエンティティだけが知っている秘密からハッシュを発生するために使われる。両方で同じハッシュ・アルゴリズムを実施する。送信された結果が予期された応答と同じであれば、ピア

10

20

30

40

50

は秘密を知っているに違いないと次に考える。ハッシュを送信することで、秘密は安全でないリンク上で決して暴露されることはない。毎回唯一のキーを使うことにより、以前の応答は以後の認証で不正に使うことができない。

* 公開／秘密鍵－公開秘密鍵それ自体で、ピアは秘密鍵の持ち主であると証明するのに使える。証明書は、その所有権と現実の世界のアイデンティティを結びつけるのに使える。文書のハッシュを作成し、次にそのハッシュを秘密鍵で暗号化することで、文書は署名される。文書の変更はハッシュを変えるので、署名を無効にする。mSecureはメタファイルを署名するのにこの特性を使う。その結果として、スクリプト自体の中にプレーンテキストの証明書を埋め込む必要なしに、スクリプトが選ばれたアイデンティティで実行されるのを可能にする。

10

【0067】

C.5 許可とポリシー

mSecureは管理する各アイデンティティに対し暗号化された記憶場所を提供する。パイプ・プロセッサはアイデンティティにより署名される／書かれるので、このことは、各パイプ・プロセッサはそれ自体専用の安全な記憶場所にアクセスできることを意味する。その記憶場所内で、mSecureはアイデンティティごとにサブ記憶場所を提供する。例えば、mSecureは「mrFile」パイプ・プロセッサのアイデンティティに記憶場所を提供する。mrFileは、他のアイデンティティがデータにアクセスできないこの記憶場所に許可とポリシーデータを格納できる。その保管スペースに、mrFileは「ジョン・ドウ」や「アニー・ウォン」のアイデンティティのため、個別のデータを格納できる。mrFileは「ジョン・ドウ」に対する「設定／許可」をチェックできる。スクリプトがmrFileアイデンティティで署名されていれば、この記憶場所にアクセスできる。このことは、スクリプトをパイプ・プロセッサの許可とポリシーに関する管理および構成の一部として使うことができることを意味する。この暗号化された記憶場所に格納できるデータは、階層化されたネーム・バリュー構造になっている。ネーム・バリューの各対は各アイデンティティに対して異なるバリューを格納できる。フォルダとサブファイルを持つソース・コントロール・システムと、比較されるかもしれない。さらに、各ファイルにはさまざまなバージョンがあるかもしれない。ネーム・バリューの対は、ユニコード・テキストである。その記憶場所は生のバイナリ・データをサポートしない。

20

【0068】

例えばパイプ・プロセッサがアンインストールされる場合のように、アイデンティティがシステムから取り除かれる場合、関係する記憶場所は自動的に削除される。記憶場所は、暗号化された形式でファイルシステムの中に存続する。これらファイルの暗号化は、安全なハードウェアに保管されたキーを使用することで一層強固に保護される。これが行われると、万一ファイルシステムが破壊され、データが別の機器に転送されても、安全なハードウェア（つまりSIMカード）がないので、データを読むことができないことを意味する。

30

このレベルのシステム・サポートを持って、コア・パイプ・プロセッサはそれらの振る舞いをきめ細かく管理できる。例えば、mrFileはファイルシステムの異なるエリアに対して読み出し／書き込みのアクセス管理ができる。

40

【0069】

C.6 ユーザ設定用UI

機器上のアイデンティティと関連データ管理するため、mrixはmrixConfigと呼ばれるユーザ・インタフェース・コンポーネントを含んでいる。このユーザ・インタフェースは以下のことを可能にする。

- ・任意の認証データを持つ（ユーザ）アイデンティティの追加／削除
- ・スクリプトおよびパイプ・プロセッサ（それ自体がアイデンティティ）の追加／削除
- ・上記に対する許可とポリシーの授与と取り消し

ユーザ設定ツールが使用可能になる前に、ユーザは自分自身を認証しなければならない、ということは驚くにあたらないはずだ。このことは、個人使用及び企業使用の両方にお

50

いて携帯電話機の共同利用を可能にする。換言すると、個人で電話機を購入したら、その許可のサブセットを企業のITアイデンティティのために使用させることができる。そのITアイデンティティは、企業のアプリケーションとデータをインストールしたり削除したりするのに使われる。個人がその後これら企業のアプリケーションやデータを管理できるかどうかは、個人と企業との「契約」しだいである。mrixは様々なオプションを選択できるようにしている。このケースでは、個人が望むなら、電話機からそのようなアプリケーションとデータを追放する選択肢をいつでも持っている。代わりに、企業がモバイル機器を所有して従業員に貸し出し、個人のアプリケーションをインストールし実行する許可を与えることもある。前と同じように、このデータは個人の私有のものなので、企業が望むなら追放することができる。そもそも企業が個人に許可を与えているのだから。電話機のそのような共有は、アイデンティティが他のアイデンティティに許可を授与および委任することを可能とする。通常はこの共有には財務上の同意があるかもしれないが、今回はmrixの範囲外である。そうは言っても、mrixによって提供される設備は、有料のサービスを実装するのに使うことができる。

10

20

30

40

50

【0070】

C.7 mrixコアのインストール

mrixコアは、ROMに焼き付けられるか、電話機購入後追加コンポーネントとしてインストールされる。初期状態では、mrixは「ファクトリ・デフォルト」と呼ばれるアイデンティティを使用する。このアイデンティティは、企業IT部門やソニー・オンライン・ゲームのような、電話機を管理するための追加のユーザ・アイデンティティを作成するために使

【0071】

C.8 第三者のインストール

インストール用のフィルの場所は、mrixコアに関しては決まっている。しかし、Identity.iniファイルは、新しいパイプ・プロセッサのdllごとに修正される必要があることに留意。スクリプトは現在「可能」許可を持たないので、すべてのスクリプトは誰かにより呼び出される。そのような呼び出しの行為は、使用するパイプ・プロセッサの振る舞いに明らかに左右される。その結果、スクリプトが使用するアイデンティティに依存する。スクリプトの実行または試験を除いて、スクリプトが任意のアイデンティティに対して期待されているように実行されるかを知る実際的な方法はない。最終的に、どのユーザも、パイプ・プロセッサのライセンスまたは電話機の所有権（換言すると、業務用に雇用主から従業員に貸し出された電話機）にかかわらず、電話機上にどのようなパイプ・プロセッサでもインストールできる。

【0072】

付録 1

MRIX-始めようガイド

1.1 MRIX概要

mrixはプラットフォームに寛容なワイヤレス・ネットワーク・オペレーティング・システムである。スマートフォン、パソコン、サーバの各機器上および各機器間の両方について、プラットフォームをまたがる幅広いモバイル・アプリケーションの迅速な開発を可能にするため企画された。コマンドラインで操作するツールの強力なセットであり、スクリプト言語を使用する込み入ったPCアプリケーション上に構築され、それと一体化される。さらに、mrixは、スマートフォン自体の上で実行可能なアプリケーションを書くのに使用できる。

【0073】

図2は、あり得るmrixアーキテクチャを示す図である。

【0074】

mrixは多くの要素から構成されており、それら要素はローカルリンク（IR、ブルートゥース、USB）上で、および遠隔中継（TCP/IP、GPRS）経由で、コマンドを実行するために使われる。それについては、「エラー！参照ソースが見つからない」の中で説明している

。

【0075】

アーキテクチャには、幾つかの主要要素がある。

- ・ m-Router：ベアラ接続エージェント。m-RouterはたくさんのPCおよびスマートフォン両方のコンポーネントから構成されている。IrDA、ブルートゥース、USB、シリアル等各種のショートリンク・ベアラを媒体として、スマートフォンとPC間の通信を可能にする。
- ・ リレー：リレーのmrElayD（「D」はデーモン(daemon)を表す）が、GPRS経由でPCからスマートフォンへのリモート・アクセスを可能にする。PCとスマートフォンの両方が、両者間の通信を行うためにそのリレーに接続する。
- ・ アイデンティティ・サーバ：すべてのコマンドは、遠近にかかわらず、「アイデンティティ」（人またはシステム）のために実行される。異なるアイデンティティは、異なる結果をもたらすコマンドを実行するために設定される。
- ・ ブートサーバ：スマートフォンの再起動時に開始されるmrixイベントを処理する。
- ・ コマンド・インタプリタ：コマンド・インタプリタ・モジュールのrshdは、スマートフォン上で実行され、普通は起動時に開始するように設定される。
- ・ コマンド・シェル：コマンド・シェルmrcmdは、PC上で実行される。このシェルは現在Windows（登録商標）上で実行されるが、すぐにLinuxとMac OS X上でも使用できるようになる。プログラムとスクリプトは、スマートフォン上のmrixコンポーネントと通信しやりとりするPCのために書くことができる。
- ・ Luaスクリプティングエンジン：Lua(<http://www.lua.org>)で書かれたスクリプトは、スマートフォン上で実行できる。多くの役に立つスクリプト、例えば、SMTPやFTPクライアントが公開され提供されている。
- ・ パイプ・プロセッサ：個別のスマートフォン・モジュールで、様々なスマートフォン機能の利用機会を提供するため、mrixコマンド環境を通じてアクセス可能である。

【0076】

1.2 必要条件

mrixの使用には、以下のハードウェアとソフトウェアが必要である。

- ・ IrDAまたはブルートゥースをサポートしているPC
- ・ Microsoft Windows（登録商標） 2000または以降
- ・ m-Router
- ・ mrix
- ・ スマートフォン（ノキア社製 7650、3650、6600、N-Gage、ソニー・エリクソン社製 P800）

1.3 MRIXの使用

1.3.1 m-Router—スマートフォンへの接続

PC上でコマンド・プロンプトを開き、以下をタイプする。

```
>mrouter -h
```

このコマンドはm-Routerのヘルプを表示する。コマンドはすべてヘルプ・オプションを持っており、-hまたは長い形式の-helpによって呼び出せる。

【0077】

接続するスマートフォンを探すために、以下をタイプする。

```
>mrouter -c search-devices
```

このコマンド・オプションは、近所にあるすべてのブルートゥース機器を捜す。

【0078】

列挙されている最初の4列は、順に、機器を表すために使われる任意の順序を表す記載番号、UID（スマートフォン機器に対しては、これはIMEI番号である。この例では機器8だけに表示されている）、ブルートゥースのアドレス、ブルートゥースの愛称（機器のユーザにより付与される）。

【0079】

得られた機器一覧から自分のスマートフォンを探し、次にコマンド・プロンプトで以下

のコマンドをタイプしてスマートフォンに接続する。

```
>mrouter -c connect-device -d <ブルートゥース機器名>
```

例えば、スマートフォンのブルートゥース名がNokia7650なら、コマンドは以下のようになる。

```
>mrouter -c connect-device -d Nokia7650
```

システム・トレイの中のm-Routerアイコンのスクリーンが、赤から青に変わるのが見える。

開発目的では、「search-devices」から得られる序数を使うのが最も便利だと気づくかもしれない。様々なアドレス方式を使いスマートフォンに接続できる。「-d」オプションは、<スキーマ>:<ID>の形式をとる。スキーマは、N、IMEI、BTADDR、BTNAME、ANYのどれか1つである。提示されないなら、スキーマはANYであると仮定される。Nは、list-devicesまたはsearch-devicesに対し返される各機器の隣にある記載番号に一致する。IMEIはUID欄と一致する。BTADDRはブルートゥース・アドレスに一致する。BTNAMEは、機器のブルートゥースの愛称に一致する。ANYは上記の全部にマッチする。それゆえ、様々な方法で機器に接続することが可能である。

```
>mrouter -c connect-device -d8
```

```
>mrouter -c connect-device -d IMEI :xxxxxxxxxxx
```

```
>mrouter -c connect-device -d BTADDR :xxxxxxxxxxx
```

```
>mrouter -c connect-device -d SJC xxxxxxxxxxx
```

スマートフォンとの接続を切断するために、以下をタイプする

```
>mrouter -c disconnect-device -d <ブルートゥース機器名>
```

次のようにタイプしてもよい。

```
>mrouter -c disconnect-device -d.
```

ここでピリオドは、現在接続されている機器または現在2以上の機器が接続されているなら最初に接続された機器を表す。

【0080】

1.3.2 mrcmd—PCから電話機を管理

mrcmdはPCサイドのプログラムで、スマートフォン上のパイプ・プロセッサースクリプトを実行できるようにする。スマートフォン上でパイプ・プロセッサースクリプトを実行する前に、mrix設定に対する必要なセキュリティ・レベルを設定する必要がある。これは、mrcmd環境変数を設定することで行われる。現在、アイデンティティ設定情報は、スマートフォンの\system\mrix\identity.iniファイルに格納されている。CT0アイデンティティが、パスワードGOODでこのファイルの中に設定されている。mrixシステムをいじくるにはこのアイデンティティを使うべきである。これは、DOSコマンド・シェルから以下のようにタイプすることで実施できる。

```
>set mrcmd=-i CT0 -a GOOD
```

代わりに、これを継続的に設定することを望むときは以下のようにする。

デスクトップ上の「マイコンピュータ」を右クリックし、「プロパティ」を選ぶ。

「詳細設定」タブを選ぶ。

「環境変数」ボタンをクリック。

「システム環境変数」リストの中の「新規」ボタンをクリック。

「変数名」欄に「MRCMD」、「変数値」欄に「-i CT0 -a GOOD」と記入。

変更を保存するため、OKを3回クリック。

【0081】

一旦セキュリティを設定したら、スマートフォン上のリモート・シェル・デーモンrshdを開始する必要がある。スマートフォン上で最初にmrixを実行するときだけ、これをしなければならない。その後は、rshdはmrixブートサーバを使い起動時に自動的に開始される。rshdを実行するためには、スマートフォン上でmrixアプリケーションを開き、リストボックスの最初のコマンドを実行する必要がある。そのコマンドは、以下のはずだ。

```
mrtcp
```

10

20

30

40

50

スクリプトは、コマンドラインで「Hello, World!」と表示する。この方法によれば、ス 50

```
#!luarun
```

50

```
-- 現在のアイデンティティ、この場合はゲストを保存
old_id = mrix.getcurrentidentity()
-- ユーザ名CT0を使い新しいアイデンティティ・ハンドルを作成する
new_id = mrix.makenewidentity("CT0","GOOD")
-- この新しく作成されたアイデンティティを使い、次のコマンドを実行する
mrix.setcurrentidentity(new_id)
mrix.write("hello, world!\n")
-- mrpromptパイプ・プロセッサを実行する
-- mrix.runは他のスクリプトとパイプ・プロセッサを実行する。mrix.runの書式は決ま
    っている。
-- mrix.run(コマンド, コマンドパラメータ, [オプションの入力])
res = mrix.run("mrprompt",-t YESNO -p "\"Need help?\"")
mrix.write("Result =".. res.."\\n");
-- 保存されているアイデンティティを復活させる
mrix.setcurrentidentity(old_id)
-- リソースを使える状態にするため新しいアイデンティティを開放する
mrix.releaseidentity(new_id)
>>>>>>>>>>>>>>>>>>
```

付録 2

mrix 2.0 カーネル内部

2 概論

mrix v2 カーネルは、それ自体のプロセス／スレッド内でSymbian「サーバ」として動作する。クライアントはMrixClient経由でカーネルを呼ぶ。カーネル自体は、さらに小さなモジュールからできている。パイプ、バンドル、ストリング、リンクのような基本的で原始的なオブジェクトが、MrixKernelObjectsと呼ばれるモジュールの中に収容されている。別のモジュールMrixKernelIdentityが、アイデンティティ／ポリシー・サービスを提供するため、これらのクラスの上に組み立てられる。三番目のモジュールMrixKernelTasksは、dll、exe、scriptにより提供されるタスクを、作成し実行する能力を付加する。最後に、MrixKernelが、前述のモジュールを駆動し調和よく組み合わせる、実際のSymbianサーバとセッション・クラスを実装する。これらのモジュールはすべて、現在のところdllである。小さく実行可能なスタブ(MrixKernelLoader)がこれらdllをロードし、プロセス空間でそれらを実行する。カーネルのはるか将来のバージョンでは、個別のdllが破壊されるのを防ぐために、これらのモジュールすべてを1つの実行ファイルにまとめる必要があるかもしれない。カーネルのセキュリティの多くは、様々なクラスを組み合わせることによりもたらされている。ある程度は、全モジュールが理解されたときにだけ、全体の図式が明らかになるが、各モジュールが個別に果たす役割を理解することは可能なはずである。うまくいけば、このことは、（可能なところではエクスポートされるAPIを同一に保って）他の部分に与える影響がもしあったとしてもごくわずかで、各部分の保守を可能にする。

【 0 0 8 9 】

カーネルのクラス構造（図4参照）は、一見したところ複雑に見えるかもしれないが、（できれば）予想より簡単であるように、パターンとレイヤーの繰り返しからできている。このクラス構造の一部を、図5に示す。

【 0 0 9 0 】

3 MrixKernelObjects.dll

テキストファイル・ログ収集機能を提供するユーティリティ・クラス「CUtil_Logger」を除いて、クラスの大部分は概念的なMSharedObjectBaseクラスから派生しているかそれに関係している。

【 0 0 9 1 】

3.1 ベースクラス

MsharedObjectBaseクラスのデフォルトの実装は、CKern_ObjectBaseが提供する。これは、その中にCManagerと呼ばれる関連するファクトリクラスを持っている。他のクラスは、繰返す実装パターンを提供するためこれらのクラスから派生される。CKern_ObjectBaseは、参照計数方法を提供するが、参照計数が0になっても、オブジェクトは直ぐには削除されない。そうではなくて、マネージャー・クラスのアクティブオブジェクトが、無効なオブジェクトを非同期ですべてゴミ収集する。削除を遅らせることには、いくつかの利点がある。

- ・参照計数がゼロになるストリングのようなクラスは、実際に削除される前に調べられ、必要なら「救助され」、参照計数が増やされる。

- ・ある機能の開始時にオブジェクトを指すポインターが有効なら、削除は非同期で行われるので、その機能の実行中に呼び出された別の機能が開放しても、その機能の終了までそのポインターは有効である。これは、mstreamの中でのアクセス違反の原因だった。そのような問題を避けるようプログラムすることは可能である（場所によっては実施されている）が、プログラムが少し複雑に見える。本仕組みは、最初からそのようなことが起こらないようにして問題を回避している。それは、循環依存の問題も単純化する。オブジェクトが開放されるとき、それが参照するオブジェクトを開放する。直ぐに削除される（そうして、プログラムの問題を引起す可能性を作る）より、このオブジェクトが完全に自分を切り離し削除されてから、オブジェクトは削除される。

- ・カーネルの呼び出しを、より速く実行できる。メモリの解放と圧縮が、クライアントが始めた操作に同期して行われるのではなくて、アイドル時間に処理されるからである。

- * オブジェクト作成時におけるクリーンアップスタックの呼び出しを回避する。従来は、ある方法が去れば削除されるのを知っていて、新しいオブジェクトを気前よくクリーンアップスタックに押し込んでいる。遅らせた参照計数仕組みの1側面は、オブジェクトは参照計数ゼロで作られる。何かオブジェクトの（部分的な）所有権を得なければ、後でゴミ収集される。クリーンアップスタックと不要物除去装置は素晴らしいが、計測可能なほどの性能上のオーバーヘッドを招くので、必要とされるところだけで使われる。

【0092】

ファクトリ／子供のパターンは、いくつかの派生するクラスで利用されている。例えば、CKern_Stringのマネージャーは、その子供を順序付きのリストに配列し、そのリストの検索を可能にする。これは、ストリングの共有可能な辞書を提供する。この1つの利点は、「c:\systemlibs\mrix\mrFile.dll」および「ゲスト」のような共通ストリングは、一旦蓄積され関与しているクラス間で共有される。別の例では、CKern_Linkマネージャーが、セッションで使われる全オブジェクトを追跡し認可することを可能にする（詳細は後述）。

CKern_LibraryおよびCKern_Authenticatorのような他のモジュールの中にあるクラス自体は、もっと専門化したクラスを提供するために、サブクラスを派生する。現在のところ、これらサブクラス（ライブラリ・タイプおよびオーセンティケータタイプ）は、カーネルの一部である。外部または第三者のサブクラスを許可するメカニズムをカーネルに加えることは可能だが、明らかにセキュリティに影響を及ぼす（カーネル・メモリ・アドレス空間にロードされた第三者コード）。

クラスにより提供される操作は、本質的に非同期である。概念的なベース・クラスMSharedAsyncRequestが、そのような操作をカプセル化するために使われる。CancelRequestバッチャル法は、そのような要求の取り消しを可能にする。

MSharedObjectBaseから派生するクラスのインスタンスのすべては、カーネルの中のそのオブジェクトを独自に識別する「ハンドル」を持っている。現在このハンドルは実際のところ「この」ポインター（唯一のものなので）だが、セキュリティの理由からカーネルの将来バージョンではおそらく変わるだろうから、このことを当然のことと決めてかかるべきでない。ハンドルがわかると、CManagerがそれに適合するハンドルを持つインスタンスを捜すために使われる。こうして、ハンドルをポインターに投げないで、ポインターをハンドルに投げるだけである。そうは言っても、後述するCKern_Linkオブジェクトは、この

10

20

30

40

50

ハンドル（）値を集めることをオブジェクトに委任する。

CManagerから派生しているオブジェクトにより所有されるオブジェクトには、テンプレート化された反復器を使うことにより反復適用される。この反復器は所有されている全オブジェクトに踏み込み、それぞれに対して求められるインタフェースを提供することを試みる。そのインタフェースが提供できないなら、反復器は静かに次のオブジェクトに移動する。ハンドルをポインターに投げることはない。

オブジェクトは、カーネルの異なる部分により共有される。上に述べたようにハンドルを参照することにより、カーネルのクライアントがオブジェクトを共有できる。ハンドルによるオブジェクトの公開と共有は、MrixKernelObjectsの中に実装されているCKern_Linkオブジェクトが提供するフラグ（アクセス権ビット）を使い、MrixKernel.dllにより行われる。

10

【0093】

3.2 オブジェクトクラス

本セクションは、MrixKernelObjectsが提供する原始的なクラスの目的と様相を、簡単に説明する。

【0094】

3.2.1 パイプ

パイプはバイナリ・バイトのFIFO(First-In First-Out)である。パイプには両端がある。リーダエンドとライタエンドである。ライタエンドに書き込まれたバイトは、ある程度はパイプの内部に保留される。次にこのバイトは書き込まれたのと同じ順番でパイプの外に読み出される。ライタはパイプを「閉じる」ことができる。このことは、それ以上のデータをそのパイプの中に書き込むことが許されないことを意味する。パイプが空になり、リーダがエラー(EOF)を返すまで、依然としてバイトは閉じられたパイプから読み出される。

20

データはパイプの中に書き込まれるので、書き込まれるバイトの計算が続けられる。概念的には、これがパイプの長さである。パイプが閉じられているときに、リーダはこの値を取り出すことができ、すでに読み出した量を知っているので、読み出しが必要な量を知ることができる。例えば、長さが既知のファイルを書き込む場合、ライタは前もってこの「パイプ長」を指定できる。そのときは、合計でそのデータ量を超えてパイプに書き込むことは許されない（その「長さ」に達した時点で、パイプは自動的に閉じられる）。パイプの長さが知られる前に、リーダがパイプの長さを要求すると、値-1が戻される。

30

このパイプ長は、パイプの実装がその中に保留することができるデータの量ではない。パイプが内部に保留することができる量は4kだったが、今は256バイトである。しかし、クライアントはバッファサイズを決めてかかるべきでない。パイプに対するバッファサイズを減らしたことは、パイプ当たりのメモリ（そして実効的にmrixカーネルのメモリ）が以前の量の1/16であることを意味する。バッファサイズが減らされると、小さな塊でデータを転送するためIPC呼がもっと必要になり、遅れを増加する。256バイトは妥当な数字に見える。「十分に速く」かつ「多すぎる量のメモリ」を消費しない。現在パイプに対する循環バッファは、カーネルのかたまりの中から割当てられている。そのようなバッファは小さいが、将来の改善は「チャンク」を使用することだろう。これはSymbianが提供するメモリの割当てであり、mrixカーネルのかたまりの中から割当ててのではなく、グローバル・メモリ・プールから割当てる。

40

クラスの実装は、循環バッファにデータを書き込んだり読み出したりするためにMSharedAsynchRequestオブジェクトを使う。これがどのように働くにかについては、例で説明するほうがわかりやすい。クライアントがパイプのライタエンドに16kのデータを書く要求をする。パイプの実装はクライアントからのデータで256バイトの循環バッファを満杯にするが、クライアントの要求をまだ完了していない。この時の前後いずれかに、別のクライアントがパイプから8kの読み出しを試みる。パイプは読み出すクライアントのためにできるだけたくさん（この場合は256バイト）書き出す。しかし、クライアントの要求をまだ完了しない。循環バッファを書き込みクライアントが補充することができるか調べるが、

50

この場合はできるので、書き込みクライアントから循環バッファに別の256バイトが読み込まれる。次に読み出しクライアントにこれを書き出す。ライタが転送するデータがなくなるか、読み出しクライアントのバッファが満杯になるかのどちらかになるまで、このループを繰り返す。このケースでは、読み出しクライアントのバッファが最初に一杯になり、その時点でその非同期要求は完了する。さらに16kのデータを読むため別の要求を行い、最終的にはライタのデータは使い果たされる。その時点でクライアントの非同期要求は完了する。パイプのコードは、基本的に2つのクライアント（プロセス／スレッド）間でデータを動かす。256バイトの塊ごとにコンテキスト・スイッチを必要としないので、比較的簡単に素早くこれができる。コンテキスト・スイッチは悪いので（つまり、時間がかかり、スピードを落とす）、避けるのがたいていは良い考えだ。読み出しクライアントは、「1つまたはもっと」読むと要求することができる。その場合、できるだけ多くのデータを転送するとき、パイプ・オブジェクトは読み出し非同期要求を完了する（つまり、循環バッファが空になり、リーダに相当量のデータを転送している場合）。クライアントは要求を取り消すことができ、その場合は直ぐに処理される。パイプのリーダエンドまたはライタエンドに要求を追加または要求を取去ることは、パイプ転送エンジンの非同期「起動」の引き金となるので、エンジンがたとえ止まっても（それは決して起こらないし、起こるべきでない）、次の要求がされる／取り消されるときに再起動される。

10

パイプはバイト・ストリームなので、ユニコードのストリングをそのまま転送することはできない。UTF8（この良い特徴は西洋文字はまだ読める）として知られるユニコード・テキストのコード化がある。カーネルはユニコード・データを受入れ、utf8に変換し、それをパイプに書き込むことができる。メモリのオーバーヘッドを抑えるために、これを大量に行う。

20

【0095】

3.2.2 ストリング

大部分のプログラムがストリングを使うが、理由は異なる。時には、ストリングはファイル名またはフィールド名のようにプログラムの一部であり、またある時は、ユーザ・インタフェース・メッセージである。ストリングは時には8ビットで、別の時には16ビットである。16ビット・ストリングの使用は、明らかに8ビット・ストリングの2倍のメモリを使うので、適切に使うべきである。そうは言っても、ストリングは有益なもので、頻繁にカーネルから出し入れされる。多くの場合に、ストリングは激しく再利用される。例えば、「mrStorage」および「--list THING --option SOMETHING」のようなストリングは、繰り返し利用される。カーネルが内部にそのようなストリングの複製を持ち続けたら、メモリが直ぐに足りなくなってしまう。必要なメモリの量を減らすために、カーネルはストリングを共有ストリング・オブジェクトを指すポインターに変換する。CKern_StringのCManagerクラスは、全ストリングを順序づける。ストリングは二分探索法を使い探されるので、カーネルの中に例えば1024のストリングがあれば、マッチするストリングを探すのに運が悪い場合でも10回ストリングの比較をするだけだ。一旦これが行われると、ストリング・オブジェクトはそのオブジェクトを指す32ビットのポインターを比較することにより比べられる（大文字小文字の区別あり）。これは素早く行われる。

30

40

ストリングは、ストリング「ハンドル」を指定することにより、クライアントに戻すことができる（そしてたいてい戻される）。クライアントは、次にコンテンツを取って来るか、コンテンツの長さを知りたいなら尋ねることができる。カーネルはユニコードのストリングをUTF8相当に変換するので、クライアントはどちらかのフォーマットでデータを要求することができる。スレッドの境界を越えてストリングを動かすには、IPC要求の中でハンドル通過させる以上に、オーバーヘッドを持つ。従って、クライアントは、ストリングの代わりにストリング・ハンドル通過させることにより、自身を加速することができる。しかし、これをする、ソース・コードの読みやすさが犠牲になる。

【0096】

3.2.3 バンドル

50

「バンドルは不運な誤解の産物である。機会を与えれば、クリスマスには良くなっているだろう。」

バンドルは、ネーム・バリュー・ペア・リストを提供するために、カーネルの内部で激しく再利用されるデータ構造である。内部では、ストリング・オブジェクト・ペアのリストが維持されている。リストには同じバリューを持つ複数のエントリが可能である。ストリング・キー・バリューに基づき、バリューを追加／置き換え／取り去る方法がある。キーは大文字小文字の区別をする。

各ペアはそれに関連する32ビットのフラグ・ワードを持っている。これは、クライアントへのアクセスを管理するためMrixKernelにより使用される（詳細は後記）。バンドルはクライアント間で共有でき、変更を待ち受けられている。換言すれば、バンドルを「見張る」ために、クライアントは非同期の要求を行うことができる。そのバンドルが変更（バリューが変更される、項目が削除される、取り除かれる）されると、バンドルを見張っている全クライアントに通知される。クライアントは次に監視要求を再提出でき、バンドルにバリューを尋ねることができる。競合条件を避けるために、その順序で行われる。

【0097】

3.2.4 リンク

CKern_Linkオブジェクトは、他のオブジェクトを指すポインターを収容している。32ビットのフラグ・ワードを公開し、指し示すオブジェクトをまとめる。CKern_Linkの実装は、集約されたオブジェクトに委任するため、Handle()とGetExtensionInterface()を上書きする。

各クライアント・セッション（後述）は、そのセッションにより使われる全「リンク」を所有する関係するCKern_Link::CManagerインスタンスを持っている。セッションがオブジェクトにハンドルを手渡すとき、そのセッションが知っているオブジェクト全部に対して有効になる。現実には、このことは、オブジェクトがそのセッションに対するCManagerインスタンスの中にあることを確認する。同時に、タイプとアクセス権（つまり、32ビットのフラグ・ワード）は、クライアントが要求している操作に関して調べられる。セッションが開放されると、そのセッションが所有するリンクのすべても（CManagerオブジェクト経由で）開放される。セッションは1リンクについて複数の参照計数を持つことができるが、これは集約されたオブジェクトについては1つの参照計数と同等である。

【0098】

4. MrixKernelIdentities.dll

4.1 アイデンティティ

概念的には、活動はすべてアイデンティティが行い、（通常他の）アイデンティティが所有するリソース上での行為である。アイデンティティは、人間のことも、遠隔にある自動化されたプロセスのことも、アプリケーションのことも、認証されているスクリプトのこともある。アイデンティティは、それを識別するためにプログラム上で使うことができる独自の名前を持っている。アイデンティティは人間が読める他の名前または記述を持つこともできるが、本質的にはアイデンティティは名前の付いた「行為者」である。アイデンティティが参照されるとき、アイデンティティが本当に主張どおりのアイデンティティかどうかについてはある程度の確実性／不確実性がいつもつきまとう。実生活では、ある人がその主張どおりの人物であるとどのようにして証明できるだろうか。

アイデンティティが別のアイデンティティのリソース上で行動するとき、許可されていることとどのように振舞うべきかに関しポリシーがある。カーネルの中では、リソースを所有するアイデンティティはコンテキスト・アイデンティティと呼ばれる。こうして、CT0のようなアイデンティティはmrFileとの関係で、リソース上で行動できる。クライアントが明示的にそのコンテキストを設定しない限り、共有のグローバル・コンテキストが使用される。

各ライブラリ（以下で説明）はアイデンティティを持っており、現在のところこのアイデンティティはライブラリの名前を元に自動作成される。しかし、将来このアイデンティティは、ライブラリにデジタル署名でサインする結果として入手されるほうがよいだろう。

10

20

30

40

50

要求可能な特別な「ストック・オブジェクト」アイデンティティがある。

・ゲスト・ネゴシエーションの済んでいない外部のピアが、デフォルトで使えるアイデンティティ。

・DefaultUserIdentity-ユーザ・インタフェースおよび他の「ローカル」モジュールにより使用されるアイデンティティで、「全面的」許可を持つ。将来は、このアイデンティティが別のアイデンティティのための（変更可能な）エイリアスになる可能性がある。将来は、デフォルト・ユーザ・アイデンティティは、機器に対し完全な管理権限も持たなくなりそうだが、このバージョンでは持っている。このアイデンティティを要求し使用するアプリケーションとコードは、署名するか暗証番号入力により認証する必要がある可能性もある。（これは、明らかにユーザ・インタフェースの考慮事項である）。 10

・SharedGlobalContext-これは全スクリプトおよび全「mr」ライブラリにより共有されるコンテキストであるが、別のコンテキストを明示的に指定し使用することができる。

【0099】

図6に示されているような、個別のポリシー／許可の店のマトリクスを想像することができる。各店の内部にネーム・バリュー・ペア（現在バンドルとして実装されている）があるが、そのようなポリシーの店が数式または他のバイナリ・データを蓄積するのを止めるものは何もない。重要なことは、そのような論理的なポリシー／許可の店／データベースがあることを理解することである。

【0100】

図6には示されていないが、どのアイデンティティもどちらの軸上にも現れることができる。任意のアイデンティティとコンテキストのペアに対するポリシーは、存在しないかも知れない。このカーネルの実装では、ポリシーはネーム・バリュー・ペアから構成されるが、これについてもまた（必要なら）変更／進化する。カーネルの中で、CKern_Identityクラスは名前の付いたアイデンティティとそのアイデンティティが本物であるという信頼度をカプセルに包み込んでいる。そのクラス内に所有または収容されるポリシーや許可はない。それどころか、CManagerクラスがそのような情報を尋ねられる。 20

アイデンティティ・マネージャはそれ自体リソースであり、従って実際にアイデンティティ・データへの読み出しまたは書き込み許可を与えるか否かは、このデータを要求する人物のアイデンティティに依存する。つまり、以下の状況が生じる。

・要求するアイデンティティAは、アイデンティティBに関する設定はアイデンティティCのコンテキストの中にある、ということを知りたい。 30

または、

・要求するアイデンティティAは、アイデンティティCのコンテキストの中にあるアイデンティティBに対する設定を修正したい。

CManagerクラスは機能を公開しているので、それゆえそのような質問を許可する。

クライアントはコンテキスト・アイデンティティ（上記「C」）を設定でき、要求アイデンティティ（上記「A」）を設定できる（はずである）。

アイデンティティ・マネージャは、現在以下のポリシーを適用する。

・DefaultUserIdentityだけが、ポリシーの変更を要求することを許可されている。

・DefaultUserIdentityは、どのようなタスク／ライブラリを実行することも許可されている。 40

現在のところ、全「mr」ライブラリが自分のアイデンティティとしてそのコンテキストを設定しているわけではなく、デフォルトのGlobalSharedContextアイデンティティを使用している。この欠点は、例えば「mrFile」に対する設定名は、例えば「mrContracts」の設定とかち合うことができないことである。

将来は（ユーザやライブラリのような）アイデンティティを取り去ることは、そのアイデンティティを含むポリシーも削除するかもしれない（そうでなければ、機器は時間の経過とともに古く使われていないポリシーで一杯になる）。

現在のカーネルでは、ポリシー情報はバンドルとしてメモリに収容されており、開始時にテキストファイルidentity.iniから一度読み込まれる。バイナリファイルにこのデータを 50

保存し取り出すために、取るに足らないほどのコードを加えることは（再構成を考えれば）可能である。将来はそのようなポリシーデータはSymbianデータベースに蓄積される可能性もある。カーネルの未来バージョンでは、このポリシーデータの一部は、SIM（スマートカード）にさえ格納されるかもしれない。

【0101】

4.2 オーセンチケーター

アイデンティティ・オブジェクトは、名前を使用して獲得することができるが、認証プロセスの結果として獲得するのが理想的である。ユーザを認証する実際の仕組みは、アイデンティティそれ自体およびそれがどのように使われるかとは独立している。CKern_Authenticatorクラスおよびそれから派生するクラスは、ピア・アイデンティティとネゴシエーションするため一連のチャレンジとレスポンスに使用できる。

プレーンテキストまたはMDハッシュまたは証明書／署名のチャレンジであれ、認証を実施する方法は、チャレンジとレスポンスがどのように転送されるかには関係ない。場合によっては、チャレンジがないかも知れない。例えば、rshdで使用されるプレーンテキスト認証では、バイト長がゼロのチャレンジがある:)。カーネルの内部にオーセンチケーターを持つことは、カーネルの外側では利用できない（つまりパスワードでは利用できない）特権を持つアイデンティティにアクセスできることを意味する。カーネルの将来バージョンでは、オーセンチケーターは認証の一部を実行するためにSIM（スマートカード）を使用するかも知れない。

現在のところ、実装されている唯一のオーセンチケーターは、プレーンテキスト・ユーザ名・パスワード・オーセンチケーターである。

明らかにネゴシエーションは、入ってくることもあり、出てゆくこともあり、対称的なこともある。オーセンチケーターを作成するときは、ピアに提示するためのローカル・アイデンティティを指定できる。ネゴシエーション（の成功）後、そのピアを代理するCKern_Entityオブジェクト（信頼水準を含む）はオーセンチケーター・オブジェクトから入手できる。

【0102】

5 MrixKernelTasks.dll

このモジュールは、タスクの実行を可能にするため、前述のモジュールとクラスの上に組み立てられる。これらのタスクは、ライブラリの内部で実行される（歴史的にパイプ・プロセッサと呼ばれる）。これらのライブラリは、多様な形態を持つdllから実行可能ファイルやスクリプトまで様々な方法で実装される。図7がそれを示す。

【0103】

5.1 タスク

CKern_Taskオブジェクトは、以下の状態の1つ状態を持つ。

- ・作成されている。
- ・開始している。
- ・実行している。
- ・止まっている（結果のコードを持って）

多数のプロパティを収容しており（バンドルとして内部に実装されている）、その中にはアイデンティティ、入出力パイプ、入出力環境を含んでいる。タスクは少なくとも1つのそれに関係するライブラリとコマンドラインのペアも持っている。スクリプトの場合には、2つ以上のペアがあってもよい。例えば、あるタスクは「myScript.lua」と「-run thing」のペアを持っているかもしれないが、「lua5.exe」と「myScript.lua-run thing」のペアも持っている。（実際のところ、ライブラリはCKern_Libraryインスタンスへのポインターであり、タスクは、ライブラリの内部で実行しているがそのライブラリのアイデンティティと、タスクを実行しているアイデンティティにアクセスできる。）

クライアントはタスクを作成し、その時点で様々なシステム・リソースが割当てられる。次に、タスクが開始される。内部では、これによりタスク・オブジェクトは開始可能と実際に印をつける。ある時点で、ライブラリはこのタスクの要求を満たし、終了コードを付

けてタスクを停止／完了する。タスクを作成したクライアントは、タスクが開始や終了するのを非同期的に待ってもよい。

タスクを実行するとき、実行するために渡されるアイデンティティは、以下の1つでなければならない。

- ・ストック・アイデンティティ
- ・それ自身のためにライブラリの内部から回収されたアイデンティティ
- ・認証されているアイデンティティ

換言すると、CT0アイデンティティと単に主張することはできず、タスクを実行するときにはそれを証明しなければならない。（既知のパスワードを使ってでも）

5.2 ライブラリ

ライブラリはタスクの列を持っている（CKern_Task::CManagerとして実装されている）。由来するクラスの実装しだいで、ライブラリはそのタスクを処理する進行中あるいは停止中のコードを実行する。

多様な形態を持つdllの場合には、これにはワーカ・スレッドの中にそのdllをロードすることを含む。多態なdllはカーネルにタスクを要求する。カーネルはワーカ・スレッドとそれを作成したライブラリを関係付け、要求があれば実行するためにそれをタスクに渡す。スレッドが不意に死ねば、その中で実行されているどのタスクも「停止」と印を付けられる。タスクが開始される前にスレッドが死ねば、実行されていないタスクは停止と印を付けられる。このことは、粗悪なライブラリがしばしば死ねば、その作成者と次に続くタスクは停止と印される（タスクの結果コードはスレッドの終了コードに設定される）。もしこれをしなかったら、常にライブラリをロードしようとして失敗する固定したループで（以前のバージョンのカーネルで起こったように）カーネルが立ち往生してしまう可能性がある。

実行可能ファイルの場合は、現在のところ派生するライブラリ・クラスが各実行タスクに対してアクティブオブジェクトを作る。そのアクティブオブジェクトは、その実行可能ファイルのインスタンスを開始する。実行可能ファイルはその「タスク」に関してカーネルに要求し、カーネルはクライアントのスレッド／プロセスを正しいライブラリ・オブジェクトおよびアクティブオブジェクトと組み合わせる。実行可能ファイルがタスクを完了／停止の設定をすることなく終了すると、アクティブオブジェクトがこの事実を検知し、その実行可能ファイルの終了コードを使いタスクは完了していると設定する。

スクリプトは、内部でその実行を別のライブラリに委任するライブラリにより処理される。その別のライブラリは同様にまた委任でき、あるライブラリがそれを処理するかエラーが発生するまで、委任を繰返せる。

最終的に、カーネル内部で実行される未使用のライブラリ・クラスが残る。それは、コマンドMrixKernelCmdの中にプログラムされた／組み立てられたハードを提供する。そのコマンドは、アイデンティティと許可を管理するために使われる。外部コマンドとして書かれることもあり得たが、そのときはこのライブラリにより使われる内部インタフェースを公開する必要があったろう。MrixKernelCmdは現在のところ未完了／テスト未実施であるが、ユーザが安全な方法で許可とポリシーを管理することができる時点で完成するのは簡単なはずだ。ライブラリはすべてクライアントと同じく振舞うように見える。

各ライブラリは関係するアイデンティティを持ち、現在そのアイデンティティはライブラリ名を元に自動作成される。しかし将来は、このアイデンティティは証明書を付けてライブラリ・バイナリにサインする結果として供給されるかもしれない。

各ライブラリは、それに関係する「情報」バンドルも持つ。現在はこのバンドルは空であるが、将来は自動作成されるデータを持つかもしれない。あるいは将来このバンドルが、ライブラリにメタデータ・ファイル供給することで、しっかりと中身を持つかもしれない。そのようなライブラリ・データは将来以下のものを含むかもしれない。

- ・著作権
- ・バージョン
- ・リソース・ストリング

10

20

30

40

50

- ・ライセンス情報
- ・ライブラリ実行のための著者／証拠
- ・アップデートするためのURL

6 MrixKernel.dll

これは、カーネルを作成するためにすべてをまとめるモジュールである。図8に示す。モジュールがロードするとき、CServer由来のクラスを作成する。次にクライアントは、CSession由来のクラスを要求できる。

Symbian IPCの変更のため、IPCの中にあったAPIの各種の仕組みが分離されていることに留意のこと。例えば、メッセージの二番目の主張を得るために、ファンクションInt32(aMessage,2)等を使用できる。これは、コードの条件付編集をモジュールのシングル部分に移す。 10

CMrixKernelServerはいくつかのマネージャーを作成し所有する。

CKern_String: CManager—これはカーネル内部で共有されるすべてのストリングの「辞書」である。

CKern_Bundle::CManagerとCKern_Pipe::CManagerは、クライアントがバンドルやパイプを作成してもらう必要があるとき、ファクトリとして使用される。

CKern_Identity::CManagerは、アイデンティティ・オブジェクトのファクトリの役を務め、またそれらのアイデンティティのポリシーを尋ねられるオブジェクトとしての機能も果たす。

CKern_Authenticator::CManagerは、要求されるタイプのオーセンチケータ・オブジェクトを作成する。マネージャーは、パスワード情報にアクセスしアイデンティティを取得および「返却」できるように、アイデンティティ・マネージャーへの照会で初期化される。 20

CKern_Library::CManagerは、ライブラリに由来するクラスを見つける／ロードする／アンロードする責任がある。

サーバの組み立て時に、これらのマネージャーはいくつかのストック・オブジェクト（つまり、ゲスト・アイデンティティ、空パイプ等）を作成するために使われる。

各クライアントは、少なくともセッションに由来するクラスを1つは作る。CMrixKernelSessionクラスはクライアントからの要求を処理する。各セッション・オブジェクトはCKern_Link::CManagerクラスのインスタンスを所有する。オブジェクトが作成され、参照ハンドルがクライアントに戻されるとき、そのリンクの使用許可を含むリンク・オブジェクトが付加（または再用）される。クライアントが要求しハンドル・バリューを渡すとき、セッション・マネージャーはそのハンドルを有効化するためリンク・マネージャーを使う。要求された操作に対して、リンクが正しいフラグ（ビット）セットを持っているかも調べる。要求が非同期であれば、その要求を処理するため（由来する）インスタンスを作成する。例を以下に挙げる。 30

E0pWritePipeの場合：

```
{
iLinks- > LogLine8 (NULL,"WritePipe");
CKern_Pipe::CWriter.
CKern_Link *link=
```

40

```
GetObjectL(aMessage,0,CKern_Pipe::CWriter::ETypeUid,(TAny*)&pWriter,mrix::K Can
Write);
```

```
CMrixKernelAsyncRequest
```

```
*req=NewRequestLC(E0pWritePipe,aMessage,3,Length(aMessage,3));
```

```
req- > AddLinkL(link);
```

```
pWriter-> QueueWriteL(req,Length(aMessage,3),Int32(aMessage,1));
```

```
CleanupStack::PopAndDestroy();//req now owned by writer
```

```
break;
```

```
}
```

50

このコードの断片は、

- ・機能が呼び出されているログ。
- ・メッセージの第0番目のパラメータで渡されるハンドルにマッチするオブジェクトを得る。CKern_pipe::CWriterタイプであることを調べる。このセッションが所有するそのオブジェクトのためのリンクがmrix::KCanWriteフラグを含むことを調べる。条件の一部でも満足されないなら、その方法はKErrAccessDeniedを残しておく。
- ・メッセージの第3番目の主張で指定されるクライアント・バッファを使用する要求オブジェクトを作成する。
- ・リンクとその要求を関係付け、そして次に
- ・非同期で完了するライタ・オブジェクトに関する方法呼び出す。
- ・ライタは参照計数を獲得しその要求を現在「所有」しているので、クリーンアップスタック上でこのコードが持っている参照を処分する。

10

【0104】

6.1 ハンドルと共有

前述のように、各セッションは（リンク・オブジェクト経由で）既知のオブジェクトのリストを維持し、これらのオブジェクトに対する許可フラグを持っている。オブジェクトを作成する以外に、オブジェクトが別のクライアントと明示的に共有されているなら、クライアントはオブジェクトへのハンドルも獲得できる。例えば、クライアントAがパイプへのハンドルを持っていれば、クライアントAが明示的にそのセッションIDを指定しそのバリューをクライアントBと共有するまで、クライアントBはそのハンドル・バリューを使用することはできない。（各クライアント・セッションは唯一のIDを持っている）。また、オブジェクトをタスクまたはバンドルに渡すことは、タスク・ハンドルが別のクライアントに（タスク実行の一部として）転送されるなら、そのタスクに収容されているオブジェクトに対する権利もまた転送される。

20

【0105】

6.2 ログ収集

mrικカーネルは強力なデバッグ用のログ収集機能を含んでいる。多分強力過ぎるので、デフォルトではデバッグ・カーネルの中にだけログが蓄積される。c:\logs\mrικKernelと呼ばれるディレクトリを作成することにより、カーネルはメモリに持っているバイナリ・パイプ・データやその他危険のないデータを始めとするオブジェクトのすべてを周期的にダンプする。言い換えると、エミュレータ上のIntuwaveの内部でデバッグするのは今のところ良いが、カーネル・セキュリティの防御を打ち負かすので、決してそのコードを大きく広い世界に出してはならない。さらにデバッグにおいて、ベース・クラスの結果としてコンポーネント・ログを作る。例えば、mrFileのために、c:\logs\mrFileと呼ばれるディレクトリを作成すると、各クライアントのインスタンスは行われたすべての呼び、結果、転送された全データをログする。以下は1例である。

30

【0106】

【表 6】

Started logging----- Completed request 0

----- Begin command

----- IsLogging

----- Completed request 0

----- End command

29BDCFE4 Created share

29BDCFE4 Set share flags 00000007

29BDBE9C Created share

29BDBE9C Set share flags 0000000B

29BDC970 Created share

29BDC970 Set share flags 0000000D

29BD984C Created share

29BD984C Set share flags 00000003

29BDCC9C Created share

29BDCC9C Set share flags 0000000D

29BD9B90 Created share

29BD9B90 Set share flags 00000007

29BDCFB8 Created share

29BDCFB8 Set share flags 00000003

29BDBB9C Created share

29BDBB9C Set share flags 00000007

----- Begin command

----- GetTaskDetails

29BDCFE4 Object passed in

29BDCFE4 LinkObject ref=1

29BDCFE4 Flags = S,R,W,

29BDCFE4 Task ref=4

29BDCFE4 Library = Z:\System\Libs\mrix\MRFILE.DLL

29BDCFE4 Params = --list

29BDCFE4 State = Running

29BDD078 Bundle ref=1

29BDD078 Key=StdIn Flags=0000000B

29BDBE9C PipeReader ref=3

29BDC418 Pipe ref=2

29BDC418 Total = -1, Written=0, Read=0

29BDC418 Start = 0, Length=0, Max=512

【 0 1 0 7 】

10

20

30

40

50

【表 7】

29BDBE9C	Accessor ref=3	
29BDC204	Accessor ref=1	
29BDD078	Key=StdOut Flags=0000000D	
29BDC970	PipeWriter ref=3	
29BDC6A8	Pipe ref=2	
29BDC6A8	Total = -1, Written=0, Read=0	
29BDC6A8	Start = 0, Length=0, Max=512	10
29BDC90C	Accessor ref=1	
29BDC970	Accessor ref=3	
29BDD078	Key=AuxIn Flags=00000003	
29BD984C	PipeReader ref=6	
29BD95E8	Pipe ref=2	
29BD95E8	Total = 0, Written=0, Read=0	
29BD95E8	Start = 0, Length=0, Max=512	
29BD984C	Accessor ref=6	20
29BD9884	Accessor ref=3	
29BDD078	Key=AuxOut Flags=0000000D	
29BDCC9C	PipeWriter ref=3	
29BDC9D4	Pipe ref=2	
29BDC9D4	Total = -1, Written=0, Read=0	
29BDC9D4	Start = 0, Length=0, Max=512	
29BDCC38	Accessor ref=1	30
29BDCC9C	Accessor ref=3	
29BDD078	Key=EnvIn Flags=00000003	
29BD9B90	Bundle ref=8	
29BDD078	Key=EnvOut Flags=00000007	
29BD9B90	Bundle ref=8	
29BDD078	Key=Param Flags=00000003	
29BDCFB8	String ref=4	
29BDCFB8	Value = --list	40
29BDD078	Key=Identity Flags=00000007	
29BDBB9C	IdentityRef ref=5	
29BDBE9C	Set share flags 0000000B	

【 0 1 0 8 】

【表 8】

29BDC970 Set share flags 0000000D
 29BD984C Set share flags 00000003
 29BDCC9C Set share flags 0000000D
 29BD9B90 Set share flags 00000003
 29BD9B90 Set share flags 00000007
 29BDCFB8 Set share flags 00000003
 29BDBB9C Set share flags 00000007
 ----- Completed request 0
 ----- End command
 ----- Begin command
 ----- GetIdentityName
 29BDBB9C Object passed in
 29BDBB9C LinkObject ref=2
 29BDBB9C Flags = S,R,W,
 29BDBB9C IdentityRef ref=5
 ----- Returning narrow string = CTO Team
 ----- Completed request 0
 ----- End command
 ----- Begin command
 ----- GetConfidenceLevel
 29BDBB9C Object passed in
 29BDBB9C LinkObject ref=2
 29BDBB9C Flags = S,R,W,
 29BDBB9C IdentityRef ref=5
 ----- Completed request 0
 ----- End command
 ----- Begin command
 ----- GetIdentitySetting
 ----- String = AllowmrFile
 29BDBB9C Object passed in
 29BDBB9C LinkObject ref=2
 29BDBB9C Flags = S,R,W,
 29BDBB9C IdentityRef ref=5

10

20

30

40

【 0 1 0 9 】

【表 9】

29BDE094 Created share

29BDE094 Set share flags 00000007

----- Returning narrow string = YES

----- Completed request 0

----- End command

----- Begin command

----- WritePipe

29BDC970 Object passed in

29BDC970 LinkObject ref=2

29BDC970 Flags = C,S,W,

29BDC970 PipeWriter ref=3

29BDC6A8 Pipe ref=2

29BDC6A8 Total = -1, Written=0, Read=0

29BDC6A8 Start = 0, Length=0, Max=512

29BDC90C Accessor ref=1

29BDC970 Accessor ref=3

----- End command

29BDC970 Async write bytes

512 bytes = 31 34 2F 31 31 2F 32 30 30 31 20 20 32 32 3A 35 34 09 20 20 20 20 20
 20 20 20 20 20 20 20 20 20 20 20 20 36 20 61 64 64 72 65 73 73 2E 64 61 74 0D 0A 32 37 2F 30 35
 2F 32 30 30 32 20 20 31 33 3A 33 37 09 20 20 20 20 20 20 20 20 20 32 31 39 36 36 37 20 42
 69 74 6D 61 70 73 2E 6C 6F 67 0D 0A 30 36 2F 30 36 2F 32 30 30 34 20 20 20 32 3A
 30 36 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 20 20 62 6C 6F 6F 6D 62 65 72 67 32
 0D 0A 30 39 2F 30 39 2F 32 30 30 32 20 20 20 34 3A 32 33 09 20 20 20 20 20 20 20 20
 20 20 20 37 39 36 20 63 6F 6E 76 65 72 74 65 64 2E 61 69 66 0D 0A 30 34 2F 31 31 2F
 32 30 30 33 20 20 20 32 3A 31 34 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 43
 6F 70 79 20 6F 66 20 74 65 73 74 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A
 30 34 09 20 20 20 20 20 20 20 20 20 20 20 20 35 35 34 20 63 75 72 72 65 6E 63 79 2E 77
 6D 6C 63 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20
 20 20 20 20 20 20 36 34 36 20 63 75 72 72 65 6E 63 79 2E 77 6D 6C 73 63 0D 0A 30 34
 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 32 37 36
 35 20 64 61 6C 69 2E 6A 70 67 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30
 34 09 20 20 20 20 20 20 20 20 20 20 20 20 32 34 32 20 64 61 74 61 2E 77 6D 6C 63 0D 0A

【 0 1 1 0 】

【表 1 0】

32 39 2F 30 36 2F 32 30 30 34 20 20 31 39 3A 30 37 09 20 20 20 20 20 20 20 20 20 20
 20 33 34 20 44 62 52 65 63 6F 76 65 72 79 4C 6F 67 2E 74 78 74 0D 0A 30 34 2F 30 39
 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20 20 34 35 30 20 64
 65 63 6B 31 2E 77 6D 6C 63 0D

Ascii = 14/11/2001 22:54. 6 address.dat..27/05/2002 13:37. 219667
 Bitmaps.log..06/06/2004 2:06.<DIR> bloomberg2..09/09/2002 4:23. 796
 converted.aif..04/11/2003 2:14.<DIR> Copy of test..04/09/2002 15:04.
 554 currency.wmlc..04/09/2002 15:04. 646 currency.wmlsc..04/09/2002 15:04.
 2765 dali.jpg..04/09/2002 15:04. 242 data.wmlc..29/06/2004 19:07. 34
 DbRecoveryLog.txt..04/09/2002 15:04. 450 deck1.wmlc.
 29BDC970 Async write bytes

512 bytes = 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20
 20 20 20 20 20 20 20 20 32 36 39 20 64 65 63 6B 32 61 2E 77 6D 6C 63 0D 0A 30 34 2F
 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20 20 31 31 33
 20 64 65 63 6B 32 62 2E 77 6D 6C 63 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35
 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20 20 34 35 36 20 64 65 63 6B 33 2E 77 6D 6C
 63 0D 0A 32 33 2F 30 37 2F 32 30 30 33 20 20 32 30 3A 33 30 09 3C 44 49 52 3E 20 20
 20 20 20 20 20 20 20 20 64 6F 63 75 6D 65 6E 74 73 0D 0A 30 33 2F 30 39 2F 32 30 30
 31 20 20 31 34 3A 31 32 09 20 20 20 20 20 20 20 20 20 20 20 34 34 38 32 20 64 6F 68 2E
 77 61 76 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20
 20 20 20 20 20 20 33 33 31 20 67 65 74 43 61 70 69 74 61 6C 2E 77 6D 6C 73 63 0D 0A
 31 31 2F 30 38 2F 32 30 30 33 20 20 31 34 3A 33 34 09 3C 44 49 52 3E 20 20 20 20 20
 20 20 20 20 20 69 63 6F 6E 73 0D 0A 32 38 2F 31 32 2F 32 30 30 33 20 20 31 34 3A 33
 30 09 20 20 20 20 20 20 20 20 20 20 38 36 36 36 20 6C 69 73 74 4E 65 77 73 53 74 6F
 72 69 65 73 49 6E 53 74 6F 72 61 67 65 2E 6C 6F 67 0D 0A 32 38 2F 31 32 2F 32 30 30
 33 20 20 31 34 3A 33 30 09 20 20 20 20 20 20 20 20 20 20 20 32 30 33 35 20 6C 69 73 74 53
 74 6F 63 6B 50 72 69 63 65 73 49 6E 53 74 6F 72 61 67 65 2E 6C 6F 67 0D 0A 32 39 2F
 30 36 2F 32 30 30 34 20 20 31 38 3A 35 30 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20
 20 20 6C 6F 67 73 0D 0A 31 36 2F 31 32 2F 32 30 30 33 20 20 31 33 3A 33 33 09 20 20
 20 20 20 20 20 20 20 20 20 20 34 35

Ascii = .04/09/2002 15:04. 269 deck2a.wmlc..04/09/2002 15:04.
 113 deck2b.wmlc..04/09/2002 15:04. 456 deck3.wmlc..23/07/2003
 20:30.<DIR> documents..03/09/2001 14:12. 4482 doh.wav..04/09/2002

【 0 1 1 1】

【表 1 1】

15:04.	331 getCapital.wmlsc..11/08/2003 14:34.<DIR>	icons..28/12/2003
14:30.	8666 listNewsStoriesInStorage.log..28/12/2003 14:30.	2035
	listStockPricesInStorage.log..29/06/2004 18:50.<DIR>	logs..16/12/2003 13:33.
45		
29BDC970 Async write bytes		
	512 bytes = 20 4D 44 53 20 44 65 6D 6F 0D 0A 30 36 2F 30 37 2F 32 30 30 33 20	
	20 31 34 3A 34 35 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 6D 6D 73 74 65 73	
	74 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20	
	20 20 20 20 34 34 32 20 6D 6F 72 74 67 61 67 65 2E 77 6D 6C 63 0D 0A 30 34 2F 30	
	39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 31 36 32 20	
	6D 6F 72 74 67 61 67 65 2E 77 6D 6C 73 63 0D 0A 32 30 2F 31 30 2F 32 30 30 33 20	
	20 31 32 3A 32 30 09 20 20 20 20 20 20 20 20 20 20 20 20 20 32 33 20 6D 72 70 75 74 2E	
	74 78 74 0D 0A 32 33 2F 30 37 2F 32 30 30 33 20 20 32 30 3A 33 30 09 3C 44 49 52 3E	
	20 20 20 20 20 20 20 20 20 20 6D 73 67 74 65 73 74 0D 0A 30 34 2F 30 39 2F 32 30 30	
	32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20 32 30 35 20 4D 75 6C 74	
	69 43 61 72 64 2E 77 6D 6C 63 0D 0A 32 33 2F 30 37 2F 32 30 30 33 20 20 32 30 3A 33	
	30 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 6E 6F 6B 69 61 0D 0A 30 38 2F 30	
	39 2F 32 30 30 32 20 20 32 32 3A 33 32 09 20 20 20 20 20 20 20 20 20 20 38 33 36 20	
	6F 72 69 67 69 6E 61 6C 2E 61 69 66 0D 0A 32 30 2F 31 30 2F 32 30 30 33 20 20 31 35	
	3A 34 31 09 20 20 20 20 20 20 20 20 20 20 20 20 31 31 39 20 72 64 61 74 2E 74 78 74 0D	
	0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20	
	20 20 34 32 39 20 72 65 61 64 6D 65 2E 77 6D 6C 63 0D 0A 31 38 2F 30 33 2F 32 30	
	30 31 20 20 31 31 3A 34 36 09 20 20 20 20 20 20 20 20 32 35 33 34 34 30 30 20 72 67 62 5F	
	71 63 69 66 0D 0A 30 34 2F 30 39 2F 32 30	
	Ascii = MDS Demo..06/07/2003 14:45.<DIR>	mmstest..04/09/2002
15:04.	442 mortgage.wmlc..04/09/2002 15:04.	162
	mortgage.wmlsc..20/10/2003 12:20.	23 mrput.txt..23/07/2003 20:30.<DIR>
	msgtest..04/09/2002 15:04.	205 MultiCard.wmlc..23/07/2003 20:30.<DIR>
	nokia..08/09/2002 22:32.	836 original.aif..20/10/2003 15:41.
	rdat.txt..04/09/2002 15:04.	429 readme.wmlc..18/03/2001 11:46.
	rgb_qcif..04/09/20	2534400
29BDC970 Async write bytes		

【 0 1 1 2 】

512 bytes = 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 31 38 30
35 20 73 6D 6F 6B 65 2E 77 6D 6C 63 0D 0A 32 30 2F 31 30 2F 32 30 30 33 20 20 31
35 3A 30 38 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 73 79 73 74 65 6D 0D 0A
30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20
33 32 36 20 74 61 62 6C 65 2E 77 6D 6C 63 0D 0A 33 31 2F 31 32 2F 32 30 30 33 20 20
20 39 3A 35 38 09 3C 44 49 52 3E 20 20 20 20 20 20 20 20 20 20 74 65 73 74 0D 0A 33
30 2F 30 37 2F 32 30 30 33 20 20 31 34 3A 34 31 09 20 20 20 20 20 20 20 20 20 20 37 33 37
32 38 20 74 65 73 74 2E 64 61 74 61 0D 0A 32 39 2F 30 39 2F 32 30 30 33 20 20 31 31
3A 33 30 09 20 20 20 20 20 20 20 20 20 20 20 20 31 36 34 20 74 65 73 74 2E 74 78 74 0D
0A 32 31 2F 30 39 2F 32 30 30 33 20 20 31 35 3A 33 37 09 20 20 20 20 20 20 20 20 20
20 20 31 32 37 20 74 65 73 74 32 2E 74 78 74 0D 0A 32 38 2F 31 32 2F 32 30 30 33 20
20 31 34 3A 33 30 09 20 20 20 20 20 20 20 20 20 20 20 20 39 31 39 20 77 61 74 63 68 46 6F
72 53 74 6F 72 61 67 65 43 68 61 6E 67 65 73 2E 6C 6F 67 0D 0A 30 36 2F 30 35 2F 32
30 30 32 20 20 31 33 3A 30 33 09 20 20 20 20 20 20 20 20 20 20 20 20 20 20 30 20 77 62 6C
6F 67 67 65 72 2E 6C 6F 67 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34
09 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20
20 20 20 31 32 30 20 57 65 6C 63 6F 6D 65 31 2E 77 6D 6C 63 0D 0A 30 34 2F 30 39
2F 32 30 30 32 20 20 31 35 3A 30

```

Ascii = 02.15:04.      1805 smoke.wmlc..20/10/2003 15:08.<DIR>
system..04/09/2002 15:04.      326 table.wmlc..31/12/2003 9:58.<DIR>
test..30/07/2003 14:41.      73728 test.data..29/09/2003 11:30.      164
test.txt..21/09/2003 15:37.      127 test2.txt..28/12/2003 14:30.      919
watchForStorageChanges.log..06/05/2002 13:03.      0 wblogger.log..04/09/2002
15:04.      306 weather.wmlc..04/09/2002 15:04.      120
Welcome1.wmlc..04/09/2002 15:0
29BDC970 Async write bytes

```

216 bytes = 34 09 20 20 20 20 20 20 20 20 20 20 20 31 32 34 20 57 65 6C 63 6F
6D 65 32 2E 77 6D 6C 63 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34
09 20 20 20 20 20 20 20 20 20 20 20 20 31 33 33 20 57 65 6C 63 6F 6D 65 33 2E 77 6D 6C
63 0D 0A 30 34 2F 30 39 2F 32 30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20
20 20 20 20 37 35 38 20 77 69 6E 64 65 78 2E 77 6D 6C 63 0D 0A 30 34 2F 30 39 2F 32
30 30 32 20 20 31 35 3A 30 34 09 20 20 20 20 20 20 20 20 20 20 20 34 33 32 20 77 69 6E

【 0 1 1 3 】

【表 1 3】

64 65 78 2E 77 6D 6C 73 63 0D 0A 31 38 2F 30 33 2F 32 30 30 31 20 20 31 31 3A 34 36
 09 20 20 20 20 20 20 20 32 35 33 34 34 30 30 20 79 75 76 5F 71 63 69 66 0D 0A

Ascii = 4. 124 Welcome2.wmlc..04/09/2002 15:04. 133

Welcome3.wmlc..04/09/2002 15:04. 758 windex.wmlc..04/09/2002 15:04.

432 windex.wmlsc..18/03/2001 11:46. 2534400 yuv_qcif..

29BDE0EC Async completed request 0

----- Begin command

----- StopTask

29BDCFE4 Object passed in

29BDCFE4 LinkObject ref=1

29BDCFE4 Flags = S,R,W,

29BDCFE4 Task ref=2

29BDCFE4 Library = Z:\System\Libs\mrx\MRFILE.DLL

29BDCFE4 Params = --list

29BDCFE4 State = Running

29BDD078 Bundle ref=1

29BDD078 Key=StdIn Flags=0000000B

29BDBE9C PipeReader ref=2

29BDC418 Pipe ref=2

29BDC418 Total = -1, Written=0, Read=0

29BDC418 Start = 0, Length=0, Max=512

29BDBE9C Accessor ref=2

29BDC204 Accessor ref=1

29BDD078 Key=StdOut Flags=0000000D

29BDC970 PipeWriter ref=2

29BDC6A8 Pipe ref=2

29BDC6A8 Total = -1, Written=2264, Read=2048

29BDC6A8 Start = 0, Length=216, Max=512

29BDC90C Accessor ref=1

29BDC970 Accessor ref=2

29BDD078 Key=AuxIn Flags=00000003

29BD984C PipeReader ref=5

29BD95E8 Pipe ref=2

【 0 1 1 4】

10

20

30

40

【表 1 4】

29BD95E8	Total = 0, Written=0, Read=0
29BD95E8	Start = 0, Length=0, Max=512
29BD984C	Accessor ref=5
29BD9884	Accessor ref=3
29BDD078	Key=AuxOut Flags=0000000D
29BDCC9C	PipeWriter ref=2
29BDC9D4	Pipe ref=2
29BDC9D4	Total = -1, Written=0, Read=0
29BDC9D4	Start = 0, Length=0, Max=512
29BDCC38	Accessor ref=1
29BDCC9C	Accessor ref=2
29BDD078	Key=EnvIn Flags=00000003
29BD9B90	Bundle ref=7
29BDD078	Key=EnvOut Flags=00000007
29BD9B90	Bundle ref=7
29BDD078	Key=Param Flags=00000003
29BDCFB8	String ref=3
29BDCFB8	Value = --list
29BDD078	Key=Identity Flags=00000007
29BDBB9C	IdentityRef ref=4
-----	Completed request 0
-----	End command
-----	Begin command
-----	CancelWritePipe
29BDC970	Object passed in
29BDC970	LinkObject ref=2
29BDC970	Flags = C,S,W,
29BDC970	PipeWriter ref=1
29BDC6A8	Pipe ref=2
29BDC6A8	Total = -1, Written=2264, Read=2264
29BDC6A8	Start = 0, Length=0, Max=512
29BDC90C	Accessor ref=1
29BDC970	Accessor ref=1

10

20

30

40

【 0 1 1 5 】

【表 1 5】

-----	Completed request 0	
-----	End command	
-----	Begin command	
-----	StopTask	
29BDCFE4	Object passed in	
29BDCFE4	LinkObject ref=1	
29BDCFE4	Flags = S,R,W,	10
29BDCFE4	Task ref=2	
29BDCFE4	Library = Z:\System\Libs\mrix\MRFILE.DLL	
29BDCFE4	Params = --list	
29BDCFE4	State = Terminated, Result = 0(00000000)	
-----	Completed request 0	
-----	End command	
29BDCFE4	Destroyed share	
29BDBE9C	Destroyed share	20
29BDC970	Destroyed share	
29BD984C	Destroyed share	
29BDCC9C	Destroyed share	
29BD9B90	Destroyed share	
29BDCFB8	Destroyed share	
29BDBB9C	Destroyed share	
29BDE094	Destroyed share	30
-----	Closing	

【0 1 1 6】

これは大量のデバッグ作業だが、クライアントとカーネルとのやりとりのすべてをフルダンプしている。カーネルのデバッグにとっても役立つが、部内では「mr」ライブラリがデバッグを必要とするときもそれを使う。

【0 1 1 7】

7 MrixClient.dll

カーネルのクライアント・インタフェースを公開するこのクラスは、mrixのユーザを対象とする別の文書に一番適切に記述されている。そのクラスはカーネルに接続しようと試みる。必要なら、（機器上のMrixKernelLoader.exeを使い）それを開始し、そうして、ユーザが様々なmrix機能にアクセスするのに使用可能な各種の機能を提供する。 40

【図面の簡単な説明】

【0 1 1 8】

【図 1】携帯機器上のリソースへのセキュア・マルチエンティティ・アクセスを提供するシステム（「mrix」）の構成例で、迅速なアプリケーション開発用の構成例を示す図である。

【図 2】本発明の実装に適しているmrixアーキテクチャを示す図である。

【図 3】mrixがどのようにたくさんの要素から構成されるかを示す図である。要素はローカルリンク（IR、ブルートゥース、USB）上や遠隔中継（TCP/IP、GPRS）経由でコマンド 50

を実行するために使われる。

【図 4】 mrixカーネルのクラス構造を示す図である。

【図 5】 図4のクラス構造の一部を示す図である。

【図 6】 個別のポリシー／許可の店のマトリクスを示す図である。

【図 7】 MrixKernelTasks.dllのクラスを示す図である。

【図 8】 MrixKernel.dllのクラスを示す図である。

【図 1】

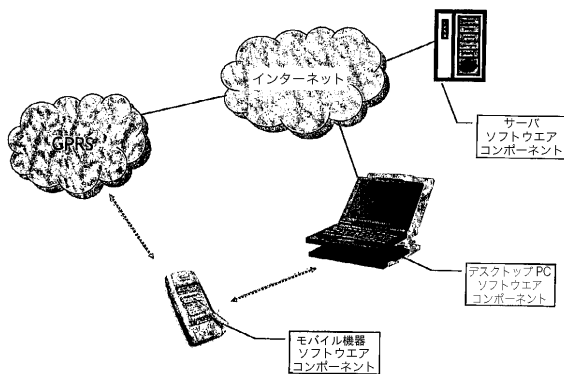


Figure 1

【図 2】

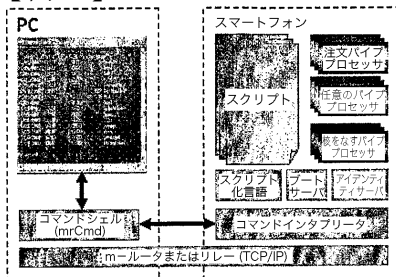


Figure 2

【図 3】

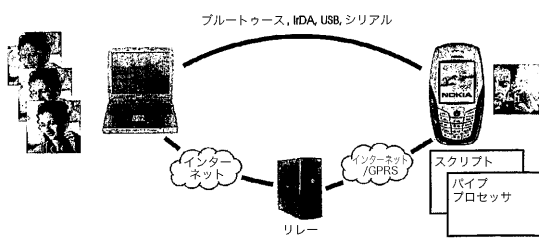


Figure 3

【図 4】

Figure 4

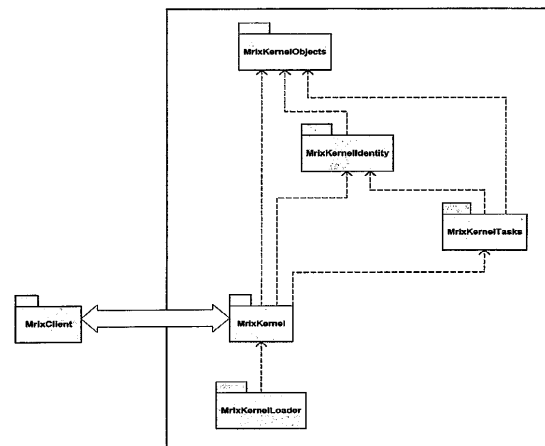


Figure 5

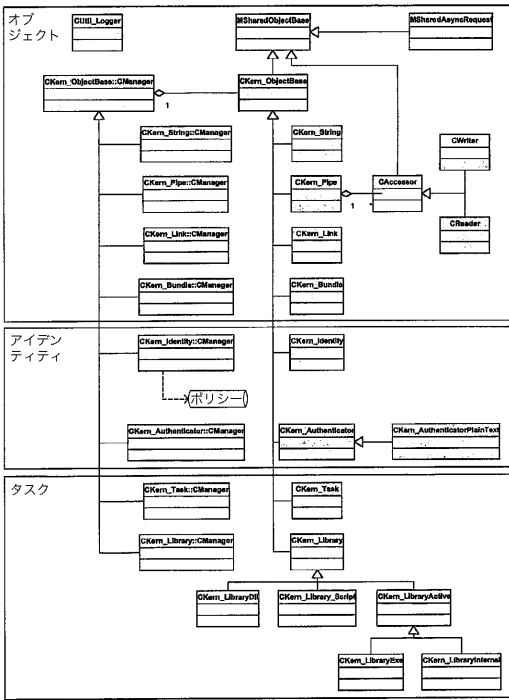


Figure 6

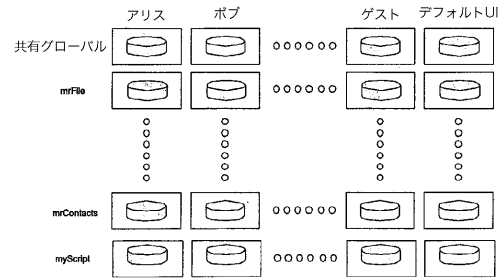


Figure 7

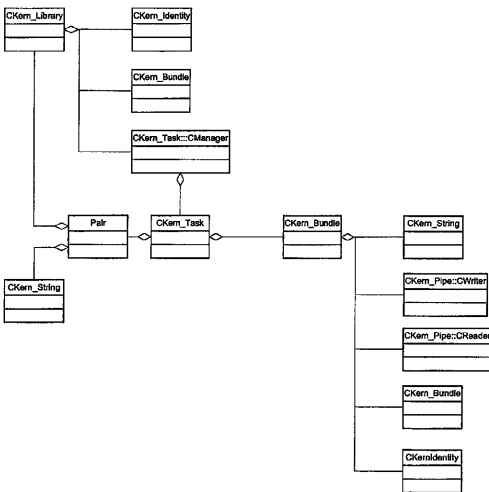
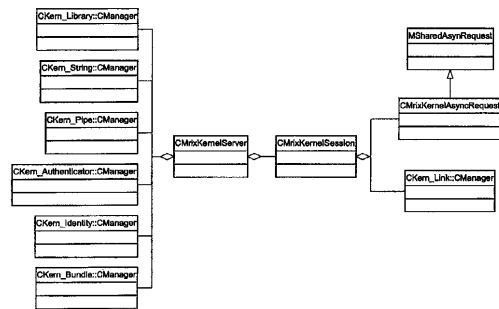


Figure 8



【国際調査報告】

INTERNATIONAL SEARCH REPORT		Int. Application No. PCT/GB2004/004701
A. CLASSIFICATION OF SUBJECT MATTER IPC 7 H04Q7/32 G06F9/46 H04L29/06		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) IPC 7 H04L G06F H04Q		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practical, search terms used) EPO-Internal, WPI Data, PAJ		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/041154 A1 (TRAN TRUNG M) 27 February 2003 (2003-02-27) paragraphs '0002!, '0004! paragraphs '0010!, '0011! paragraph '0021! paragraph '0032! paragraphs '0035!, '0037! paragraph '0041! paragraphs '0044!, '0045! ----- -/-	1, 3-6, 20-38
☒ Further documents are listed in the continuation of box C. ☒ Patent family members are listed in annex.		
* Special categories of cited documents: *A* document defining the general state of the art which is not considered to be of particular relevance *E* earlier document but published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *Z* document member of the same patent family		
Date of the actual completion of the international search 10 January 2005		Date of mailing of the international search report 19/01/2005
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel: (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016		Authorized officer Ruiz Sanchez, J

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/GB2004/004701

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JSR 118 EXPERT GROUP: "Mobile Information Device Profile for Java <TM> 2 Micro Edition, version 2" JCP SPECIFICATION. JAVA 2 PLATFORM, MICRO EDITION, MOTOROLA INC. AND SUN MICROSYSTEMS INC., 5 November 2002 (2002-11-05), pages 1-10, 23-34, 501-513, XP002312736 page 5 - page 9 page 23 - page 27 page 29 - page 30 page 32, last paragraph page 502 page 504, last paragraph - page 505 page 512	1-38
Y	WO 99/44137 A (SUN MICROSYSTEMS INC) 2 September 1999 (1999-09-02) page 3, lines 12-15 page 4, line 20 - page 6, line 20 page 9, lines 19-30 page 11, lines 8-14 page 12, line 23 - page 13, line 26 page 14, line 29 - page 17, line 31	1-38
Y	LAI C ET AL: "User authentication and authorization in the Java<TM> platform" COMPUTER SECURITY APPLICATIONS CONFERENCE, 1999. (ACSAC '99). PROCEEDINGS. 15TH ANNUAL PHOENIX, AZ, USA 6-10 DEC. 1999, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 6 December 1999 (1999-12-06), pages 285-290, XP010368605 ISBN: 0-7695-0346-2 the whole document	1-8, 16, 17, 20-35, 37, 38

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No.

PCT/JP2004/004701

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 2003041154	A1	27-02-2003	NONE	
WO 9944137	A	02-09-1999	US 6138238 A	24-10-2000
			AU 2680299 A	15-09-1999
			AU 2680399 A	15-09-1999
			AU 2680499 A	15-09-1999
			AU 2686699 A	15-09-1999
			AU 2686799 A	15-09-1999
			AU 2766199 A	15-09-1999
			AU 2769899 A	15-09-1999
			AU 2770199 A	15-09-1999
			AU 2770299 A	15-09-1999
			AU 2770399 A	15-09-1999
			AU 2770499 A	15-09-1999
			AU 2770599 A	15-09-1999
			AU 2787699 A	15-09-1999
			AU 2787799 A	15-09-1999
			AU 2787899 A	15-09-1999
			AU 2876899 A	15-09-1999
			AU 2876999 A	15-09-1999
			AU 2878399 A	15-09-1999
			AU 2878499 A	15-09-1999
			AU 3297199 A	15-09-1999
			AU 3297299 A	15-09-1999
			AU 3300499 A	15-09-1999
			AU 3300599 A	15-09-1999
			AU 3309199 A	15-09-1999
			CN 1292115 T	18-04-2001
			CN 1292116 T	18-04-2001
			CN 1292113 T	18-04-2001
			CN 1292117 T	18-04-2001
			CN 1292192 T	18-04-2001
			CN 1292118 T	18-04-2001
			CN 1298502 T	06-06-2001
			CN 1298503 T	06-06-2001
			CN 1298504 T	06-06-2001
			CN 1298505 T	06-06-2001
			CN 1298523 T	06-06-2001
			CN 1298524 T	06-06-2001
			CN 1298506 T	06-06-2001
			CN 1298507 T	06-06-2001
			CN 1298508 T	06-06-2001
			CN 1298509 T	06-06-2001
			CN 1298510 T	06-06-2001
			CN 1298511 T	06-06-2001
			CN 1298512 T	06-06-2001
			CN 1298513 T	06-06-2001
			CN 1298514 T	06-06-2001
			CN 1298515 T	06-06-2001
			CN 1298516 T	06-06-2001
			CN 1298525 T	06-06-2001
			DE 69903711 D1	05-12-2002

フロントページの続き

(81)指定国 AP(BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), EA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), EP(AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OA(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG), AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW

(特許庁注：以下のものは登録商標)

1. Linux
2. Macintosh
3. ウィンドウズ
4. リナックス

(74)代理人 100115071

弁理士 大塚 康弘

(74)代理人 100116894

弁理士 木村 秀二

(74)代理人 100130409

弁理士 下山 治

(72)発明者 スプーナー, デイヴィッド

イギリス国 パークシャー エスエル6 1 エヌジェイ, メイデンヘッド, ザ ブロードウェイ, シエナ コート (番地なし), インテュウェーブ リミテッド 内

Fターム(参考) 5B017 AA07 BA06 BB02 BB06

5K027 AA11 BB09 HH14 HH23

5K067 AA30 BB04 BB34 DD17 DD27 HH24

5K201 AA09 BC23 BD06 CB01 CB04 CB20 ED05 EE03 EE09