

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第5132852号
(P5132852)

(45) 発行日 平成25年1月30日(2013. 1. 30)

(24) 登録日 平成24年11月16日(2012. 11. 16)

(51) Int.Cl.

F I

G06F 9/54 (2006.01)
G06F 9/46 (2006.01)
G06F 21/12 (2013.01)
G06F 12/14 (2006.01)
G06K 19/07 (2006.01)

G06F 9/46 480Z
G06F 9/46 420A
G06F 21/22 112B
G06F 12/14 510A
G06K 19/00 N

請求項の数 13 (全 52 頁)

(21) 出願番号 特願2000-595236 (P2000-595236)
(86) (22) 出願日 平成12年1月20日(2000. 1. 20)
(65) 公表番号 特表2003-522987 (P2003-522987A)
(43) 公表日 平成15年7月29日(2003. 7. 29)
(86) 国際出願番号 PCT/US2000/001235
(87) 国際公開番号 W02000/043876
(87) 国際公開日 平成12年7月27日(2000. 7. 27)
審査請求日 平成19年1月17日(2007. 1. 17)
(31) 優先権主張番号 09/235, 157
(32) 優先日 平成11年1月22日(1999. 1. 22)
(33) 優先権主張国 米国 (US)

前置審査

(73) 特許権者 597004720
オラクル・アメリカ・インコーポレイテッド
アメリカ合衆国、94065 カリフォルニア州、レッドウッド・ショアーズ、オラクル・パークウェイ、500
(74) 代理人 110001195
特許業務法人深見特許事務所
(72) 発明者 サッサー ジョシュア
アメリカ合衆国 カリフォルニア州 94114 サン フランシスコ セヴンティーンズ ストリート #11 4150

最終頁に続く

(54) 【発明の名称】 小面積装置において入力点オブジェクトを使用してコンテキスト障壁を横断するアクセスを許可する技術

(57) 【特許請求の範囲】

【請求項 1】

プログラム・モジュールが実行される処理マシン(410)を含む小面積装置(400)を操作する方法において、

第1のコンテキスト(770)の中で、前記第1のコンテキスト(770)が所有している1つ以上のプログラム・モジュールを前記処理マシン(410)上で実行するステップと、

第2のコンテキスト(760)の中で、前記第2のコンテキスト(760)が所有している1つ以上のプログラム・モジュールを前記処理マシン(410)上で実行するステップであって、前記1つ以上のプログラム・モジュールは前記第2のコンテキスト(760)の中で入口点オブジェクト(1210)を生成するものであるステップと、

前記処理マシン(410)上で実行されているコンテキスト障壁(600、600')によって前記第1及び第2のコンテキスト(770、760)を分離して隔離するステップと、

前記実行されているコンテキスト障壁(600、600')によって、前記第1及び第2のコンテキストの内の1つの中で実行されている前記プログラム・モジュールの前記第1及び第2のコンテキストの内の他の1つの中の前記プログラム・モジュールへのアクセスを制御するステップとを備え、

前記制御するステップは、

前記実行されているコンテキスト障壁(600、600')によって、前記アクセス

が認証されていなければ、前記第1及び第2のコンテキストの内の前記1つの中で実行されているプログラム・モジュールによる前記第1及び第2のコンテキストの内の前記他の1つの中のプログラム・モジュールへの前記アクセスを防止するステップと、

前記実行されているコンテキスト障壁（600、600'）によって、前記アクセスが認証されていれば、前記第1及び第2のコンテキストの内の前記1つの中の前記実行されているプログラム・モジュールによる前記第1及び第2のコンテキストの内の前記他の1つの中の前記プログラム・モジュールへの前記アクセスを可能にするステップとを含み、

前記方法は、さらに、

前記第1のコンテキスト（770）の中で前記処理マシン（410）上で実行されているプログラム・モジュールから、前記コンテキスト障壁（600、600'）を越えて、前記第2のコンテキストの中の前記入口点オブジェクトに直接アクセスするステップであって、前記直接アクセスは前記入口点オブジェクト（1210）によって制御され、前記コンテキスト障壁（600、600'）によっては制御されないものであるステップと、を有する

ことを特徴とする方法。

【請求項2】

前記入口点オブジェクト（1210）がフィールド及びメソッドを含み、前記方法は、前記直接アクセスが前記入口点オブジェクト（1210）の前記メソッドの1つへのアクセスである場合に、前記実行している入口点オブジェクト（1210）による前記コンテキスト障壁（600、600'）を越えた前記直接アクセスを許容するステップと、

前記直接アクセスが前記入口点オブジェクト（1210）の前記フィールドの1つへのアクセスである場合に、前記実行している入口点オブジェクト（1210）による前記コンテキスト障壁（600、600'）を越えた前記直接アクセスを防止するステップと、を更に含むことを特徴とする請求項1に記載の方法。

【請求項3】

入口点オブジェクト（1210）への前記アクセスが少なくとも1つのセキュリティ検査を含む請求項1又は2に記載の方法。

【請求項4】

各コンテキストに対してそれぞれ別の名前空間を割当ててをさらに含む請求項1ないし3のいずれかに記載の方法。

【請求項5】

各コンテキストに対してそれぞれメモリ空間を割当ててをさらに含む請求項1ないし3のいずれかに記載の方法。

【請求項6】

前記処理マシン（410）がプロセッサ（300）上で実行されている仮想マシン（720）を含む請求項1ないし5のいずれかに記載の方法。

【請求項7】

請求項1ないし6のいずれか1つに記載された方法に従い小面積装置（400）を操作するためのプログラム命令を記憶したコンピュータ読み取り可能記録媒体。

【請求項8】

プログラム・モジュールが実行される処理マシン（410）を含む小面積装置（400）において、

第1のコンテキスト（770）の中で、前記第1のコンテキスト（770）が所有している1つ以上のプログラム・モジュールのグループを実行する手段と、

第2のコンテキスト（760）の中で、前記第2のコンテキスト（760）が所有している1つ以上のプログラム・モジュールを実行する手段であって、前記1つ以上のプログラム・モジュールの少なくとも1つを実行して前記第2のコンテキストの中で入口点オブジェクト（1210）を生成するものである手段と、

前記処理マシン（410）上で実行されている、前記第1及び第2のコンテキスト（7

10

20

30

40

50

70, 760) を分離して隔離するためのコンテキスト障壁 (600, 600') と、を
備え、

前記コンテキスト障壁 (600, 600') は、

前記実行されているコンテキスト障壁が前記1つのコンテキストの中で実行されている
前記プログラム・モジュールの前記他の1つのコンテキストの中の前記プログラム・モジ
ュールへのアクセスを制御するための手段を含み、

前記制御するための手段は、

アクセスが認証されていなければ、前記第1及び第2のコンテキストの内の1つの中
で実行されているプログラム・モジュールによる前記第1及び第2のコンテキストの内の
他の1つの中のプログラム・モジュールへの前記アクセスを防止するための手段と、

前記アクセスが認証されていれば、前記第1及び第2のコンテキストの内の1つの中
の前記実行されているプログラム・モジュールによる前記第1及び第2のコンテキストの
内の他の1つの中の前記プログラム・モジュールへの前記アクセスを可能にするための手
段とを有し、

前記入口点オブジェクト (1210) は、前記第1のコンテキスト (770) の中で前
記処理マシン (410) 上で実行されているプログラム・モジュールによる、前記コンテ
キスト障壁 (600, 600') を越えて、前記第2のコンテキストの中の前記入口点オ
ブジェクトへの直接アクセスを制御するように構成されており、前記直接アクセスは前記
コンテキスト障壁 (600, 600') によっては制御されないものであることを特徴と
する小面積装置。

【請求項9】

前記入口点オブジェクト (1210) は、フィールド及びメソッドを含み、前記第1の
コンテキスト (770) の中で前記処理マシン (410) 上で実行されているプログラム
・モジュールによる、前記コンテキスト障壁 (600, 600') を越えた、前記第2の
コンテキストの中の前記入口点オブジェクトへの前記直接アクセスは、前記直接アクセス
が前記入口点オブジェクト (1210) のメソッドの1つへのアクセスである場合には可
能にされ、前記直接アクセスは、前記直接アクセスが前記入口点オブジェクト (1210)
の前記フィールドの1つへのアクセスである場合には防止されることを特徴とする請求
項8に記載の小面積装置。

【請求項10】

入口点オブジェクト (1210) への前記アクセスが少なくとも1つのセキュリティ検
査を含む請求項8又は9に記載の小面積装置。

【請求項11】

各コンテキストに対してそれぞれ別の名前空間を割当てる手段をさらに含む請求項8な
いし10のいずれかに記載の小面積装置。

【請求項12】

各コンテキストに対してそれぞれメモリ空間を割当てる手段をさらに含む請求項8ない
し11のいずれかに記載の小面積装置。

【請求項13】

前記処理マシン (410) がプロセッサ (300) 上で実行されている仮想マシン (7
20) を含む請求項8ないし12のいずれかに記載の小面積装置。

【発明の詳細な説明】

【0001】

関連出願の相互参照

本出願は、1997年4月15日に出願された米国特許出願シリアル番号08/839.
621、発明の名称「セキュリティ分配バイト・コード確証を有する仮想マシン」、発明
者モシュ・レビイとジュデイ・シュワブ (ドケット番号50253-221/P3263
) の関連出願である。

本出願は、1999年1月22日に出願された米国特許出願シリアル番号09/235.
158、発明の名称「小面積装置においてコンテキスト障壁を使用してセキュリティを実

10

20

30

40

50

施する技術」、発明者ジョシュア・スーサーとミシェル・ビー・バットラー及びアンデイ・ストレイヒ（ドケット番号50253-216/P3708）の関連出願でもある。

本出願は、1999年1月22日に出願された米国特許出願シリアル番号09/235,155、発明の名称「小面積装置において実行時環境特権を使用してコンテキスト障壁を横断するアクセスを許可する技術」、発明者ジョシュア・スーサーとミシェル・ビー・バットラー及びアンデイ・ストレイヒ（ドケット番号50253-218/P3710）の関連出願でもある。

本出願は、1999年1月22日に出願された米国特許出願シリアル番号09/235,156、発明の名称「小面積装置においてグローバル・データ構造を使用してコンテキスト障壁を横断するアクセスを許可する技術」、発明者ジョシュア・スーサーとミシェル・ビー・バットラー及びアンデイ・ストレイヒ（ドケット番号50253-219/P3711）の関連出願でもある。

10

本出願は、1999年1月22日に出願された米国特許出願シリアル番号09/235,159、発明の名称「小面積装置において共有可能インターフェイスを使用してコンテキスト障壁を横断するアクセスを許可する技術」、発明者ジョシュア・スーサーとミシェル・ビー・バットラー及びアンデイ・ストレイヒ（ドケット番号50253-220/P3712）の関連出願でもある。

【0002】

発明の背景

発明の属する技術分野

20

本発明は、コンピュータ・セキュリティに関し、より詳細には、スマート・カードなどの小面積装置にセキュリティを施す技術に関する。

【0003】

従来の技術

いくつかのオブジェクト指向プログラミング言語が、当業者に良く知られている。これらの例は、C++言語及びスモールトーク言語を含む。

【0004】

別のこのようなオブジェクト言語は、ジャバ（J A V A）（登録商標）言語である。この言語は、アディソン・ウェズレイから出版されたジェームス・ゴスリング等による本、「ジャバ（商標）言語仕様」に記載されている。ジャバ（商標）言語は特に、ジャバ（商標）仮想マシン上で動作するのに適している。このようなマシンが同じくアディソン・ウェズレイから出版されたティム・リンドホルム及びフランク・イエリンによる本、「ジャバ（商標）仮想マシン」に記載されている。

30

【0005】

いくつかの小面積装置が当業者には良く知られている。これらの小面積装置は、スマート・カード、セルラー・ホン、及びその他の小面積又はミニアチュアの装置を含む。

【0006】

スマート・カードは、形と大きさがクレジットカードに似ているが、典型的に、カード中にデータ処理機能及びそれを介してスマート・カードとプログラム、データ、その他の通信を達成する1組の接点を含む。典型的に、1組の接点は、クロック入力、リセット入力及びそれを介してデータ通信が達成できるデータ・ポートと、電源接続と、戻りを含む。

40

【0007】

カード受入装置を使用して、情報がスマート・カードに書込まれ、そしてスマート・カードから読み出される。カード受入装置は典型的に、ホスト・コンピュータに取付けられた周辺装置であり、スマート・カードがその中に挿入できるスロットなどのカード・ポートを含む。一旦、挿入されると、コネクタからの接点又はブラシがスマート・カードの表面接続領域に当接して、電源を供給し、そして、スマート・カード上に典型的に見出されるプロセッサ及びメモリとの通信を許容する。

【0008】

50

スマート・カード及びカード受入装置（C A D）は、広範囲な標準化努力、例えば、I S O 7 8 1 6、の主題である。

【0009】

ファイヤ・ウォールを使用して、認証されたユーザを認証されていないユーザから分離することは、ネットワーク環境では良く知られている。このようなファイヤ・ウォールは、発明者デビッド・ブランウェルの名前で1998年12月1日に出願された米国特許出願シリアル番号09/203,719、名称「認証されたファイヤ・ウォール通過フレームワーク」（ドケット番号50435-023/P2789/TJC）に記載されている。

【0010】

完全なジャバ（商標）プラットホームの能力の小組が、スマート・カードなどの小面積装置のために定義されている。この小組は、ジャバ・カード（商標）プラットホームと呼ばれている。ジャバ・カード（商標）プラットホームの使用は以下の刊行物に記載されている。

【0011】

ジャバ・カード（商標）2.0—言語セット及び仮想マシン仕様

ジャバ・カード（商標）2.1—アプリケーション・プログラミング・インターフェイス

ジャバ・カード（商標）2.0—プログラミング・コンセプト

ジャバ・カード（商標）アプレット開発者ガイド

I S O 7 8 1 6—部分11の作業ドラフトが、批評を受けるため回覧されている。そのドラフトは、スマート・カード上で動作するための別の実行コンテキストを可能にする標準を特定している。

【0012】

実行コンテキストの一般概念はコンピュータ科学では良く知られている。一般に、計算環境内で複数実行コンテキストを使用することは、各々が他から不適当な干渉無しに動作できるように、互いに異なるプログラム・モジュール又はプロセスを分離又は隔離するための方法を提供する。異なるコンテキスト間の相互作用はもしあれば、偶然というよりは熟考されて、各コンテキストの完全性を保全するために注意深く制御される。複数コンテキストの一例が、複数の仮想マシンが定義されて、各仮想マシンがそれ自身の実行コンテキストを有すメインフレームなどのより大きなハードウェア装置に見られる。別の例として、発明者デジョングの米国特許番号第5,802,519号が、スマート・カード上での複数実行テキストの使用を開示している。当業者には理解されるように、複数実行コンテキストを提供する計算環境はさらにある与えられた実行コードとそれに対応するコンテキストとを関連付けるための機構を与える必要がある。

【0013】

現在コンテキストも周知の一般概念である。複数コンテキストを支援するある計算環境は、いかなる時点において、1つのコンテキストを特に計算の能動的集中として取扱う。このコンテキストは「現在コンテキスト」と呼ばれる。その他のあるコンテキストが現在コンテキストとなるために、現在コンテキストが変化する時、「コンテキスト切換え」が発生すると言われる。当業者には理解されるように、これらの計算環境は、コンテキスト切換えを容易にするため、どのコンテキストが現在のものであるかを記録するための機構を提供する。

【0014】

従来技術において、小面積装置の世界、特に、スマート・カードの世界においては、小面積装置上で動作するコンテキスト間の相互動作は存在しなかった。各コンテキストは完全に分離されて動作し、そして異なるコンテキスト中の他のアプリケーション又はプロセスに影響を与えることなく、そのコンテキスト空間内で動作又は故障する。

【0015】

ジャバ（商標）プラットホームに用いられるセキュリティ保護の1つの層は普通、砂箱モデルと呼ばれる。信用されないコードは「砂箱」内に置かれて、完全なジャバ（商標）環

10

20

30

40

50

境又は「現実世界」にどんな損傷も与えることなく、安全に「遊ぶ」ことができる。このような環境において、ジャバ（商標）アプレットは通信をせず、しかし、各々はそれ自身の名前の空間を有する。

【0016】

いくつかのスマート・カードのオペレーティング・システムは実行コンテキストが直接に通信することを許さず、オペレーティング・システム又はサーバーを介しての通信を許す。

【0017】

問題

小面積装置上にコンピュータ・プログラム及び他の情報を置くことを試みる時、いくつかの問題が存在する。1つの大きな問題は大変に制限されたメモリ空間の存在である。これはしばしば必要な機能をメモリ空間内に与えるために非常に大きな努力を要する。

10

【0018】

小面積装置に関連した第2の問題は、異なる小面積装置の製造業者は異なるオペレーティング・システムを使用することができるという事実である。この結果、1つのオペレーティング・システムのために開発されたアプリケーションは必ずしも、異なる製造業者により製造された小面積装置にポータブルではない。

【0019】

もし、複数のプログラム・ソース（製造業者又はベンダー）からのプログラムが1つの小面積装置に適用されるならば、セキュリティが、新しいプログラムをその小面積装置にロードする時に、既存のプログラム及びデータの劣化を回避する1つの試みとしてのファクターとなる。同じ関心が、ハッカー又は悪意ある人間がプログラム及びデータをアクセスすることを防ぐ時に、存在する。

20

【0020】

スマート・カードなどの小面積装置は、分離した仮想マシンを実現するために必要な資源を有しないことは明らかである。それにもかかわらず、別個の実行コンテキスト間で厳格なセキュリティを維持することが望まれる。

【0021】

従来は、セキュリティが、スマート・カード又はその他の小面積装置上に同じソース又は既知の信頼されるソースからのアプリケーションのみがロードされることにより与えられていた。

30

【0022】

従って、プログラマーに不適當な重荷を課さない高速で効率的なピア・ツー・ピア通信を介した安全な方法のみにより選択された実行コンテキスト間のオブジェクト指向相互作用を可能にし、しかし、信用できないソースにより異なる時に書かれたアプレットの動的なローディングを容易にすることが望まれる。

【0023】

発明の開示

本発明は、1つのコンテキストを他から隔離及び分離を与えるためのコンテキスト障壁（時々、ファイヤ・ウォールと呼ばれる）を提供し、必要な時に障壁を横断して制御されたアクセスを提供することに関する。

40

【0024】

本発明によれば、例えば、各々が1又は複数のアプレットを含み、同じ論理（すなわち、仮想又は現実）マシンで実行され、互いに保護された、2つの実行コンテキストが、オブジェクト指向言語機構などの言語機構を使用して、制御された安全な方法で情報を共有できる。例えば、セキュリティはオブジェクトごとに行われる。このように、第1実行コンテキスト内のメソッドは第2実行コンテキスト内の第1オブジェクトAにアクセスすることができるが、第2実行コンテキスト内の第2オブジェクトBには選択的なベースでアクセスできない。

【0025】

50

1つの例示的な実施の形態によれば、増強されたジャバ（商標）仮想マシン（VM）は、VM内で実行コンテキストを横断するアクセスの試みについてある実行時検査を提供する。検査は、VMにより自動化され、又はVMからの支援でもってプログラマーによりコード化できる。これは言語レベル通信機構を使用して達成できる。この方法では、作られている言語を使用して他のオブジェクトへアクセスするのと同じ方法で実行テキストを横断するオブジェクトのアクセスを表現できる。これらの実行時検査は、ジャバ（商標）言語及びプラットフォームが既に提供するものを越えた防御／セキュリティの第二次元を提供する。

【0026】

これらの機構は、例えば、プログラミング・バグ（全てのコンテキストがアクセス可能であるべきでない時に、データを「パブリック（公）」（グローバル）と宣言するような）に起因するセキュリティ・ホールに対する保護を提供する。それらはまた、細かい単位で共有の制御を可能にする（共有されるべきオブジェクト及び共有されるべきアプレットの選択など）。

【0027】

本発明はまた、コンピュータ・プログラム製品及び本発明の別の観点に関する搬送波に関する。

【0028】

本発明の上述又はその他の特徴、観点、及び利点は、以下の本発明の詳細な説明を添付図面と共に参照することにより、明らかとなる。

【0029】

表記法及び命名法

以下の詳細な説明は、コンピュータ又はコンピュータ・ネットワーク上で実行されるプログラム手順の言葉で説明される。これらの手続的記述及び表現は、当業者が彼等の仕事の本旨を最も効果的に他の当業者に伝えるのに使用される手段である。

【0030】

手順は、ここでは一般に、所望の結果に導くステップの首尾一貫した順序で説明される。これらステップは物理的量の物理的操作を必要とするステップである。必ずではないが、これらの量は普通、記憶、移送、結合、比較、又はその他の操作が可能な電氣的又は磁氣的信号の形を取る。主として共通の使用のため、これらの信号は、ビット、値、要素、シンボル、文字、言葉、数など、又は同等なものとして呼ぶことが便利であることが示される。しかし、これらの全ての及び類似の言葉は、適当な物理量に関連付けられるべきであり、そしてこれらの量に適用される単なる便利なラベルにすぎないことに注意すべきである。

【0031】

さらに、実行される操作は、時々、人間の操作者により実行される精神的な操作と共通に関連付けられる加算又は比較などの言葉で呼ばれる。本発明の一部を形成するここで記述されたどんな操作においても、大部分の場合は、人間の操作者のこのような能力は必要ではなく、又は望まれない。操作は機械的操作である。本発明の操作を実行するために有用な機械は、汎用デジタル・コンピュータ又は他の計算装置を含む。

【0032】

本発明はまた、これらの操作を実行するための装置に関する。これらの装置は所望の目的のために特別に構成されてよい。又は、選択的に活性化される又はコンピュータに記憶されたコンピュータ・プログラムにより再構成される汎用コンピュータを含んでもよい。ここで説明される手続は特定のコンピュータ又は他の装置に固有的に関係するものではない。ここでの教示に従って書かれたプログラムと一緒にさまざまな汎用機械が使用でき、又は、所望のメソッドのステップを実行するためにより専門化された装置を構成することがより便利であろう。これらのマシンのさまざまについての必要な構成が、以下の記述から明らかとなる。

【0033】

詳細な説明

付録として明細書に添付したものは、未刊行のジャバ・カード実行時環境 2. 1 仕様という名称の書類の草案である。この草案書類は本発明の特定の実施の形態のさらに詳細な説明を与える。

【0034】

本発明の技術は以降、スマート・カードの例の文脈で記載されるが、例は単なる例示であり、本発明の範囲を制限するものではない。

【0035】

図 1 は、カード受入装置 110 を具備したコンピュータ 120 及びカード受入装置 110 と共に使用するスマート・カード 100 を示す図である。動作においては、スマート・カード 100 はカード受入装置 110 内に挿入されて、そして、スマート・カード 100 の表面上でアクセス可能な 1 組の接点 105 を介して電源又はデータ接続が与えられる。カードが挿入される時、カード受入装置 110 からの嵌入接点が、表面接点 105 と相互接続してカードに電源を与え、そしてボード上のプロセッサ及びメモリ記憶との通信を可能にする。

10

【0036】

図 2 は、ネットワーク 200 に接続された、図 1 中の 120 などに示されるカード受入装置を具備したコンピュータを示す。さらに、ネットワークに接続されるのは、サーバー 210 などの複数の他の計算装置である。カードを具備した装置 120 を使用してネットワーク 200 を介してスマート・カード上にデータ及びソフトウェアをロードすることが可能である。この種のダウンロードされるものは、さまざまな電子商取引及び他のさまざまなアプリケーションに従って使用されるデジタル・キャッシュ及び他の情報、及びスマート・カード上にロードされるべきアプレット又は他のプログラムも含むことができる。カード受入装置及びスマート・カードの処理要素の制御に使用されるデータ及び命令は、揮発性又は不揮発性メモリ内に記憶することができ、若しくは、例えば、命令及び／又はデータを含む搬送波などの通信リンク上で直接に受信することが可能である。さらに、例えば、ネットワークは LAN 又はインターネットなどの WAN 又は他のネットワークであってよい。

20

【0037】

図 3 は、従来技術のスマート・カードなどの小面積装置の例示的なハードウェア・アーキテクチャである。図 3 に示すように、プロセッサ 300 は、読出し専用メモリ 315 及び／又はランダム・アクセス・メモリ 316 を含むことができる主記憶 310 と相互接続される。プロセッサはまた、EEPROM などの二次記憶 320 及びシリアル・ポートなどの入力／出力 330 に接続される。この種の小面積装置は大変に単純にできることが理解できる。

30

【0038】

図 4 は、従来技術によるプリンシパルによりアクセスされるオブジェクトを示す。図 4 に示すように、小面積装置などの物理装置 400 は、実行コンテキスト 420 を実行する 1 つ又は複数の処理マシン（仮想又は物理的）をその中に含む。実行コンテキストは、例えば、特定のアプレットと関連したコンテキストである。実行コンテキスト中の 1 つ又は複数のプリンシパル 430（例えば、アプレット又はアプリケーション）は、実行コンテキスト中の他のオブジェクトのアクセスを求めることができる。実行コンテキスト中でアクセスが発生する限り、アクセスは許可されて、全てが通常に機能するであろう。

40

【0039】

図 5 は、本発明のさまざまな実施の形態を説明するのに使用できる例示的なセキュリティ・モデルである。これは単に使用できる多くのモデルの 1 つにすぎないが、この目的には便利なモデルである。このモデルにおいて、プリンシパル（時々、エンティティと呼ばれる）は、オブジェクト 520 などの 1 つのオブジェクトについてアクション 510 を取ることを提案する。セキュリティ検査が、プリンシパル、オブジェクト、及び／又は取られることが提案されたアクション上に課せられる。

50

【0040】

図5において、2つのタイプのオブジェクトが示され、その上にプリンシパルによりアクションが取られる。これらは、データ・オブジェクト（例えば、データ1及びデータ2（520、520'））とエンティティ530を含む。プリンシパルはこれらのオブジェクトのいずれについても動作し又は動作を試みる。

【0041】

データは受動的であるが、エンティティ530は能動的である。プリンシパルから能動的エンティティへの対角線も「アクション」とラベルが付けられているが、しかし、これはデータ・オブジェクトへのアクションと比較して、機能を実行し、又は、メソッドをコールし、又は、メッセージを送信するなどのより洗練され且つ任意に複雑なアクションであることができる。データと同じく、オペレーティング・システムにより実行されるセキュリティ検査は、プリンシパルの識別、エンティティの識別、及び／又はアクションのタイプを使用することができる。さらに、能動的であるエンティティは、それ自身の追加的なセキュリティ検査を実行できる。これらは、所望ならば任意に複雑にすることができ、そしてプリンシパルの識別、エンティティ自身の識別、アクション及び／又は利用可能な他の情報も使用できる。

10

【0042】

オブジェクト指向システムにおいて（ジャバ・カード（商標）プラットフォームなど）、「オブジェクト」は典型的にデータ及びエンティティの組合せである。プリンシパルが、オブジェクトのフィールドにアクセスすることを試みる時、これはデータ・アクセサーかなり単純なセキュリティ検査により保護されるかなり単純なアクション、である。プリンシパルがオブジェクトのメソッドのアクセスを試みる時、これは、アクション及びセキュリティ検査の両方を任意に複雑できるエンティティ・アクセスである。

20

【0043】

図6は、本発明の1つの観点によるコンテキスト障壁又はファイヤ・ウォールによる実行テキストの分離を示すブロック図である。物理装置400及びマシン410は、図4に示される同じアイテムに対応する。実行コンテキスト420は、コンテキスト内でオブジェクト440にアクセスすることを試みる1つのプリンシパル430を示す。このアクセスは普通成功する。しかし、実行コンテキスト420はまた、コンテキスト障壁600を横断して、実行コンテキスト620のオブジェクト640にアクセスすることを試みるプリンシパル630を示す。普通、このアクセスはアクション635がコンテキスト障壁600を横断する所のX636に示されるように禁止される。

30

【0044】

図7は、本発明を実行するのに有用なソフトウェア・アーキテクチャを表している。このソフトウェア・アーキテクチャは実行時環境700として示されている。小面積装置についてオペレーティング・システム710が普通、使用される。本発明の例示的な実施の形態中の仮想マシン720が、オペレーティング・システム上で実現される。仮想マシンは、ジャバ・カード（商標）仮想マシン又は他の仮想マシンである。標準の仮想マシンの能力がここに記載される追加の機能を与えるために拡張できる。又は、機能は別個のモジュールとして提供できる。仮想マシン720は、インタープリタ又は実行時システム740へのアクセスを与えるネイティブ処理系730を含む。実行時システムは、オブジェクト指向処理系のオブジェクトを管理するためのオブジェクト・システム750を含む。3つのコンテキスト760、770、及び780が示される。各コンテキストが、実行コンテキスト間のコンテキスト障壁（時々、ファイヤ・ウォールと呼ばれる）により他から分離される。1つの特定の実施の形態において、コンテキスト760はスーパー・コンテキストである。すなわち、コンテキスト760は、下位のコンテキスト770及び780には利用できない特権及び能力を有する。それは入口点オブジェクト又はグローバル・データ構造を生成し、そして下位コンテキスト770及び780内のオブジェクトにアクセスする特権を潜在的に含む。

40

【0045】

50

各オブジェクトは1つの特定のコンテキストに関連付けられている。そのコンテキストはそれと関連付けられた各オブジェクトを所有すると言われる。実行時システム740は、コンテキストを独特に識別するための手段と現在実行されているコンテキスト識別し指定するための手段を与える。オブジェクト・システム750は、オブジェクトをそれらが所有するコンテキストと関連付けるための機構を与える。

【0046】

例えば、実行時740は、独特な名前でもってコンテキストを識別でき、そして、オブジェクト・システム750はコンテキストの名前をオブジェクトのヘッダー内に記録することにより、オブジェクトをコンテキストと対応して関連付けることができる。オブジェクトのヘッダー内の情報は、オブジェクト指向言語により書かれたプログラムによりアクセスできないが、仮想マシン720自身のみ利用可能である。代替的に、実行時システム740は、メモリ空間を各々が特定のコンテキストのための分離領域に分割することによりコンテキストを識別できる。そして、オブジェクト・システム750はコンテキストのメモリ空間内にオブジェクトの記憶を割当てることによりオブジェクトをそのコンテキストに対応して関連付けることができる。

10

【0047】

図8は、本発明の1つの観点によるコンテキスト障壁を実現するセキュリティ実施プロセスのフローチャートである。プリンシパルがオブジェクト(800)上のアクションを呼出すと、オブジェクトがプリンシパル(810)のコンテキスト内にあるかどうかを決定するための検査が作られる。もしなければ、アクションは不許可となる(840)。それ以外は、アクションは許可される(830)。これはコンテキスト障壁又はファイヤ・ウォールの最も単純な形である。1つの特定の実施の形態では、もしオブジェクトが名前空間の外側にあるか又はコンテキストがアクセスを要求するメモリ空間の外側にあれば、セキュリティ例外を投げることにより、アクションが不許可とされる(840)。

20

【0048】

図9は、本発明の1つの観点によるファイヤ・ウォールを横断してアクセスするオブジェクトを示すブロック図である。図9は図6と実質的に同じである。しかし、図9は、オブジェクト910上にアクション905を実行するためにオブジェクト910のアクセスを求めるプリンシパル900も示す。本発明によれば、アクション635が阻止された方法でアクセスがファイヤ・ウォール600により阻止されるというよりは、プリンシパルとオブジェクトが異なる実行コンテキスト内にあるという事実にかかわらず、プリンシパル900がオブジェクト910上のアクション905を実行できるように、アクション905がアクセス・ポイント920を介してファイヤ・ウォールを横断して生ずることを許可する。アクセス・ポイント920の背後の機構は図12-18を参照して以下に説明される。アクセス・ポイント920は、X636などの阻止されたアクセスと共に存在することに注意する。このように、アクセス・ポイント920は、コンテキスト障壁600を横断して細かい単位の制御(オブジェクト単位のセキュリティ)の共有を与える。

30

【0049】

オブジェクト・アクセス900が開始される時、現在コンテキスト設定はコンテキスト420である。もしオブジェクト910がデータ・オブジェクトであれば、アクション905は単純なデータ・アクセスであり、第2コンテキストではコードは実行されない。もしオブジェクト910がエンティティ・オブジェクトであれば、アクション905は実行されるそのオブジェクトのコードを生ずる。そのコードは第2コンテキスト620内で実行される。正しいコンテキスト620内でオブジェクト910のコードを実行するため、仮想マシン410はコンテキスト切換えを実行する。コンテキスト切換えは、現在コンテキスト設定をコンテキスト620に変えて、現在コンテキスト設定の前の値を後で再開できるように記憶する。この点から、コードは新しい現在コンテキスト内で実行される。アクション905が完了した時、制御はアクセス900後の点に戻される。復帰中、仮想マシン410は現在コンテキスト設定の値をその前の値に回復しなければならない。

40

【0050】

50

図10は、ファイヤ・ウォールを横断してアクセスするカスケードされたオブジェクトを示すブロック図である。図10は、3つの実行コンテキスト、1000、1010及び1020を示す。実行コンテキスト1内のプリンシパル1030は、実行コンテキスト2内のオブジェクト1050上のアクション1035を呼出すことを求め、コンテキスト障壁600内のアクセス・ポイント1070を介してそれを行なう。実行コンテキスト2内のオブジェクト1050は、実行コンテキスト3内のオブジェクト1060上のアクション1045の実行を求めるオブジェクト・アクセス1040を有する。それは、実行コンテキスト2及び3を分離するコンテキスト障壁600'内のアクセス・ポイント1080を使用してこれを達成する。実行コンテキスト2内のオブジェクト1050はまた、同じ実行コンテキスト内で、すなわち、実行コンテキスト2内で、オブジェクト1099上でアクション1095を呼出す別のオブジェクト・アクセス1090を有する。両方のアクション1035及び1045は図9の説明で開示されたようなコンテキスト切換えを生ずる。しかし、アクション1095はコンテキスト障壁を横断しないため、コンテキスト切換えはその実行に必要ではなく、従って発生しない。

10

【0051】

図11は、1つのコンテキスト中のプリンシパルがファイヤ・ウォールを横断して別のコンテキスト内にアクセスすることを許可するプロセスのフローチャートである。このプロセスには本質的に3つのステップが存在する。実行コンテキスト2において、アクセスされるべきオブジェクトが生成されて共有されるものとして指定される(1100)。実行コンテキスト1中において、プリンシパルが実行コンテキスト2内のオブジェクトについての参照を得る(1110)。そして、実行コンテキスト1中のプリンシパルは、コンテキスト2中で共有として指定されたオブジェクト上のアクションを呼出す(1120)。

20

【0052】

図11の項目1100中で説明したように共有可能と生成されたオブジェクトを指定又は識別することについては、本発明によれば、これはオブジェクトの表現のヘッダー内に共有可能な属性を含むことにより、行なわれる。オブジェクトのヘッダー内の情報はオブジェクト指向言語により書かれたプログラムによりアクセスすることはできないが、VM自身のみにより利用可能である。

【0053】

別のコンテキスト中のオブジェクトへの参照を獲得することは、別のコンテキスト中のオブジェクトにアクセスする特別の場合である。別のコンテキスト内の1つのオブジェクトへのアクセスを与える機構は、別のオブジェクトをまた利用可能にする。例えば、別のコンテキスト中のオブジェクト上のメソッドを呼出すことは、異なるコンテキスト中に第2のオブジェクトへの参照を戻す。追加の機構が、異なるコンテキスト中のオブジェクトへの初期参照の獲得を可能にするため必要とされる。特定の実施の形態においては、ある周知の入口点オブジェクトへの参照が公のAPIを使用して獲得できる。一旦、異なるコンテキスト内のオブジェクトに対する初期参照が獲得されると、そのオブジェクトからさらなる参照が獲得でき、さらに続けられる。

30

【0054】

本発明によれば、コンテキスト障壁を横断して情報を獲得するために、4つの一般的な手法がそんざいする。これらの手法は、コンテキスト障壁を横断してオブジェクトにアクセスするため、又は、コンテキスト障壁を横断してアクセスされるべきオブジェクトの参照を獲得するために、個別に又は組合せて使用できる(1110)。これらの手法は図12-18により説明される。

40

【0055】

図12は、コンテキスト障壁を横断してアクセスを許可するための入口点オブジェクトの使用を説明するブロック図である。図12に示すように、コンテキスト770(コンテキスト1)内のあるオブジェクト1200は、スーパーコンテキスト760内の情報にアクセスすることを望む。特定の実施の形態において、スーパーコンテキスト760は少なくとも1つの入口点オブジェクト1210を含む。入口点オブジェクト1210は、スーパ

50

ーコンテキストの下位の各コンテキストがスーパーコンテキストの入口点オブジェクトと通信できるようにするため、公のAPIの一部として刊行されるか、又は、刊行されたAPIを通じて間接的に利用可能にできる（例えば、図11を参照して前に説明した機構に従い）。（他の実施の形態では、入口点オブジェクトはスーパーコンテキスト以外のコンテキストにより収納される。）

【0056】

図13は、ファイヤ・ウォールを横断してアクセスを許可するためにグローバル・データ構造の使用を説明するブロック図である。この手法において、スーパーコンテキスト760は、グローバル配列などのグローバル・データ構造を生成する。特定の実施の形態では、スーパーコンテキスト760はこのようなグローバル・データ構造を生成することを許可された唯一のコンテキストである。（別の実施の形態では、グローバル・データはスーパーコンテキスト以外のコンテキストにより収納されてよいことが理解される。）このグローバル状態のため、コンテキスト770及び780の各々は、グローバル・データ構造を読書きすることができる。このように、1つのコンテキストによりグローバル・データ構造内に書込まれた情報は、別のコンテキストにより読み出すことができる。例えば、この機構はコンテキスト間でオブジェクトへの参照又は2進値データを送るために使用できる。

10

【0057】

図14は、コンテキスト障壁を横断してアクセスを許可するためにスーパーコンテキスト特権を使用することを説明するブロック図である。図14において、スーパーコンテキスト760におけるオブジェクトは2つに分離するコンテキスト障壁を横断してコンテキスト780へのアクセスを求める。スーパーコンテキスト760は、スーパーコンテキストに関連した特権により、コンテキスト780のどんなメソッドも呼出すことができ、そしてコンテキスト780内に含まれるどんなデータもアクセスすることができる。

20

【0058】

図15は、ファイヤ・ウォールを横断してアクセスすることを許容するため、共有可能インターフェイス・オブジェクトの使用を説明するブロック図である。共有可能インターフェイスは共有可能インターフェイス・メソッドの1組を定義する。共有可能インターフェイス・オブジェクトは共有可能インターフェイス内に定義されたメソッドの組を少なくとも実現するオブジェクトである。図15において、コンテキスト2（780）内のオブジェクト1210は共有可能インターフェイス・オブジェクトである。別のコンテキスト770中のオブジェクト・アクセス1200は、もしオブジェクト・アクセス1200のプリンシパルがオブジェクト1210自身によりそのようにすることが認証されているならば、オブジェクト1210上のどんな共有可能なインターフェイス・メソッドをも呼出すことができる。この認証は以下の図18を参照してさらに説明する。

30

【0059】

本発明による仮想マシンは、「ジャバ（商標）仮想マシン仕様」に記載される仮想マシンなどの初期の仮想マシンを越えた機能を提供することが理解される。特に、本発明によれば、仮想マシンはファイヤ・ウォールを横断したアクセスを許可するセキュリティ実施プロセスを実現又は容易にするための機能を与える。このプロセスが次に図16－18を参照して説明される。それは、上記の図12－15を参照して説明された4つの手法を限定的ではなく含む、ファイヤ・ウォールを横断するアクセスを提供するどんな手法にも適用できることに注意する。

40

【0060】

図16は、ファイヤ・ウォールを横断してアクセスを許可するセキュリティ実施プロセスのフローチャートである。プリンシパルがオブジェクト1600上のアクションの呼出しを試みる時、オブジェクトがプリンシパルのコンテキスト内にあるかどうかを決定するための検査が作成される（1610）。もしそうであるならば（1610－Y）、アクションが許可される（1630）。もしそうでなければ（1610－N）、プリンシパルによるアクションがオブジェクトに対して許可されるているかどうかを見るための検査が生成

50

される（１６２０）。もしそうであるならば（１６２０－Ｙ）、アクションが許可される（１６３０）。もしそうでなければ（１６２０－Ｎ）、アクションは許可されない。この特定の実施の形態では、セキュリティ例外が投げられる（１６４０）。

【００６１】

図１７は、ブロック１６２０のさらに詳細を示す図１６のフローチャートである。もしオブジェクトがプリンシパル（１６１０－Ｎ）のコンテキスト内になれば、プリンシパルによるアクションがオブジェクト上で許可されるかどうかを見るために複数のテスト１６２１、１６２２、１６２３…１６２９が引受けられる。これらのテストは仮想マシン・オブジェクト指向実現の中で、仮想マシンのみで、又は、仮想マシンとオブジェクトとを加えて、行なわれる。もしどんなテストにおいて合格ならば、アクションは許可される（１６３０）。しかし、もし全てのテストが否定的決定（１６２Ｘ－ノー）を生ずる場合は、アクションは許可されない。特定の実施の形態では、セキュリティ例外が投げられる（１６４０）。これらテストは図１２－１５と関連して説明された許可されたアクセスに係る。

10

【００６２】

図１８は、図１５において説明されたアクセス・メソッドに使用される図１７のブロック１６２９の例示的な実現を示すフローチャートである。８２９又は１６２９などのテストにおいて、仮想マシンはオブジェクトが共有されたオブジェクト１８１０かどうかを検査する。もしそうでなければ（１８１０－ノー）、このテストは失敗する。しかし、もしそうであれば（１８１０－イエス）、仮想マシンはオブジェクトＯ上のメソッドＡを呼出す（１８２０）。もし、オブジェクトＯ上のメソッドＡがプリンシパルが認証されていると決定すれば（１８３０）、テストは合格であり（１８４０）、そしてアクセスは許可される。それ以外は、テストは失敗である（１８５０）。これは、認証テキストをオブジェクト自身のコード内にプログラムすることを可能にする。

20

【００６３】

本発明がスマート・カードの実現と関連して説明されたが、本発明はスマート・カードに限らず、他の小面積の装置にも適用できる。小面積の装置は一般に、メモリ、計算能力又は速度において制限又は限界があるものと考えられる。このような小面積装置は、境界走査装置、フィールド・プログラム可能装置、ページャー、及びセルラー・ホンなどの多くの装置を含む。

30

【００６４】

一般に、小面積装置は、実行コンテキストの安全な相互操作に問題がある、資源の制約を受ける計算装置及びシステムである。このような小面積装置はそれらの制限された資源のためにセキュリティ手段の実現に制約が課せられる。資源の制約のため仮想マシン構成においては、単一の仮想又は物理的マシンが、複数の仮想マシンと対比して、使用されなければならない。

【００６５】

本発明はまた、本発明の特徴が有利なより大型の装置にも適用できる。例えば、本発明は、サーバレットを使用する時、もしこれらの間でオブジェクトの共有があれば、有利であることが確かめられる。いくつかのデスクトップ・システムにおいてさえも、本発明の技術が有利に使用できる。

40

【００６６】

ジャバ（商標）言語及びプラットフォームが本発明に適しているが、ある特性を有する言語又はプラットフォームが本発明を実現するのに良く適しているであろう。これらの特性は、タイプ安全性、ポインター安全性、オブジェクト指向、動的リンク、及び仮想マシンに基づくことを含む。これらの特性の全てが特定の實現において存在する必要はない。いくつかの実施の形態では、これらの特性の一つ又は複数を欠く言語又はプラットフォームが使用できる。「仮想マシン」は、ビット（仮想マシン）又はシリコン（現実／物理的マシン）のいずれかで実現できる。

【００６７】

50

本発明は、オブジェクト単位のセキュリティを示して説明されたが、クラス単位のセキュリティなどの他の手法も使用できる。

【0068】

本発明が詳細に説明されたが、説明は例示であり、限定するものではないことは明らかである。本発明の範囲と精神は特許請求の範囲の記載とその均等物のみにより限定される。

【0069】

ジャバ（商標）カード（商標）実行時環境（JCRE）2.1仕様

草案2

Copyright(c)1998 Sun Microsystems, Inc.

901 San Antonio Road, Pal Alto, CA 94303 USA

全ての権利を留保。この書類の著作権はサン・マイクロシステムズ社が所有する。

【0070】

サン・マイクロシステムズ社（SUN）は、あなたに、内部評価目的に限り使用するために、ジャバ（商標）カード（商標）実行時環境（JCRE）2.1仕様（仕様）を実施するために必要なSUNの知的所有権の非独占、移転不可、世界的、制限ライセンス（サブライセンス無し）を無料で許諾する。この制限ライセンス以外には、あなたは仕様について権利や利益を有せず、仕様を生産又は商業的使用に使用する権利を有しない。

【0071】

制限的権利標章

サンは、このソフトウェアの適当性について、非侵害性又は特定の目的の適合性又は商品可能性の黙示的保証を限定的ではなく含む、一切の明示又は黙示の保証又は表示をしない。サンはこのソフトウェア及び二次的著作物の使用、修正、又は配布の結果としてライセンスシーが受けた損害については責任を持たない。

【0072】

商標

サン、サン・ロゴ、サン・マイクロシステムズ、ジャバソフト、ジャバビーンズ、JDK、ジャバ、ジャバカード、ホットジャバ、ホットジャバ・ビュウズ、ビジュアル・ジャバ、ソラリス、NEO、ジョー、ネトラ、NFC、ONC、ONC+、オープン・ウィンドウズ、PC-NFS、エンベデッド・ジャバ、パーソナル・ジャバ、SNM、サンネット・マネージャ、ソラリス・サンバースト・デザイン、ソルステイス、サンコア、ソーラーネット、サンウェブ、サン・ワークステーション、ザ・ネットワーク・イズ・ザ・コンピュータ、ツールトーク、ウルトラ、ウルトラコンピューティング、ウルトラサーバ、ホエア・ザ・ネットワーク・イズ・ゴーイング、サン・ワークショップ、エックス・ビュー、ジャバ・ワークショップ、ジャバ・コーヒー・カップ・ロゴ、及びビジュアル・ジャバは、サン・マイクロシステムズ社の米国または他の国での商標又は登録商標である。

【0073】

この文獻は、そのまま提供されて、非侵害性又は特定の目的の適合性又は商品可能性の黙示的保証を限定的ではなく含む、明示又は黙示のいかなる種類の保証を含まない。

【0074】

この文獻は、技術的不正確さタイプ誤りを含み得、この情報への変更は定期的に行なわれる。これらの変更はこの文獻の新しい版に組み込まれる。サン・マイクロシステムズ社は、この文獻に記載されたプログラム又は製品の改良又は変更を何時でも行なうであろう。

【0075】

目次

前書

1. 導入

2. ジャバ・カード仮想マシンの一生

3. ジャバ・カード・アプレットの一生

3.1 メソッド・インストール

3.2 メソッド・セレクト

10

20

30

40

50

3. 3メソッド・プロセス	
3. 4メソッド・デセレクト	
3. 5電源喪失及びリセット	
4. 一時的オブジェクト	
4. 1一時的オブジェクトをクリアする出来事	
5. 選択	
5. 1デフォルト・アプレット	
5. 2選択命令処理	
5. 3非選択命令処理	
6. アプレット隔離及びオブジェクト共有	10
6. 1アプレット・ファイアウォール	
6. 1. 1コンテキスト及びコンテキスト切換え	
6. 1. 2オブジェクト所有	
6. 1. 3オブジェクト・アクセス	
6. 1. 4ファイアウォール保護	
6. 1. 5静的フィールド及びメソッド	
6. 2コンテキストを横断したオブジェクト・アクセス	
6. 2. 1J C R E入口点オブジェクト	
6. 2. 2グローバル配列	
6. 2. 3J C R E特権	20
6. 2. 4共有可能インターフェイス	
6. 2. 5前のコンテキストの決定	
6. 2. 6共有可能インターフェイスの詳細	
6. 2. 7共有可能インターフェイス・オブジェクトの獲得	
6. 2. 8オブジェクト・アクセス振舞い	
6. 3一時的オブジェクト及びアプレット・コンテキスト	
7. トランザクション及びアトミシテイ	
7. 1アトミシテイ	
7. 2トランザクション	
7. 3トランザクション期間	30
7. 4ネストされたトランザクション	
7. 5テスト及びリセット・トランザクション故障	
7. 6トランザクションの中止	
7. 6. 1プログラムの中止	
7. 6. 2J C R Eによる中止	
7. 6. 3J C R Eのクリーンアップ責任	
7. 7一時的オブジェクト	
7. 8コミット容量	
8. A P Iトピックス	
8. 1A P D Uクラス	40
8. 1. 1外向きデータ転送のT = 0詳細	
8. 1. 2外向きデータ転送のT = 1詳細	
8. 2セキュリティ及び暗号パッケージ	
8. 3J C S y s t e mクラス	
9. 仮想マシン・トピックス	
9. 1資源故障	
10. アプレット・インストーラ	
10. 1インストーラ	
10. 1. 1インストーラ実現	
10. 1. 2インストーラA I D	50

- 10. 1. 3 インストーラ A P D U
- 10. 1. 4 インストーラ 振舞い
- 10. 1. 5 インストーラ 特権
- 10. 2 新にインストールされた アプレット
- 10. 2. 1 インストーレイション・パラメータ
- 11. A P I 定数

【0076】

前書

ジャバ（商標）カード（商標）技術は、ジャバ・プログラミング言語の一部と、スマート・カード及び関連する小メモリ埋め込み装置に最適化された実行時環境とを結合する。ジャバカード技術の目標は、ジャバ・ソフトウェア・プログラミングの多くの利点を資源を、制約されたスマート・カードの世界へ持って来ることである。

10

この書類は、ジャバ・カード実行時環境（J C R E）2. 1 の仕様である。ジャバ・カード可能化装置のベンダーは J C R E の実現を提供する。この仕様の文脈内での J C R E の実現とは、ジャバ・カード仮想マシン（V M）、ジャバ・カード・アプリケーション・プログラミング・インターフェイス（A P I）、又はジャバ・カード技術仕様に基づいた他の部品のベンダーの実現を言う。参照実現とは、サン・マイクロシステムズ社により作られた実現である。ジャバ・カードに対して書かれたアプレットはジャバ・カード・アプレットと呼ぶ。

誰がこの仕様は使用すべきか？

20

この仕様は、ジャバ・カード技術仕様を拡張する仕様を開発、実現を作成する際、又はジャバ・カード実行時環境（J C R E）の拡張を作る際に J C R E 実現を補助することを意図している。この仕様はさらに、ジャバ・カード技術仕様の理解を深めたいジャバ・カード・アプレット開発者に対して意図している。

この仕様を読む前に

このガイドを読む前に、あなたはジャバ・プログラミング言語、ジャバ・カード技術仕様、及びスマート・カードに明るくなければならない。ジャバ技術及びジャバ・カード技術に明るくなるための良い材料が、サン・マイクロシステムズ社のウェブ・サイト、<http://java.sun.com>にある。

この仕様はどのように構成されているか

30

1 章、「J C R E の範囲と責任」は J C R E 実現に必要とされるサービスの概観を与える。

2 章、「仮想マシンの一生」は仮想マシンの一生を定義する。

3 章、「アプレットの一生」はアプレットの一生を定義する。

4 章、「短期オブジェクト」は短期オブジェクトの概観を与える。

5 章、「選択」は J C R E がどのようにアプレット選択を処理するかを説明する。

6 章、「アプレット隔離及びオブジェクト共有」はアプレット隔離とオブジェクト共有を説明する。

7 章、「トランザクションとアトミシテイ」はトランザクション中のアトミシテイの概観を与える。

40

8 章、「A P I トピックス」は J C R E に必要だがジャバ・カード 2. 1 A P I 仕様には完全に特定されていない A P I 機能を説明する。

9 章、「仮想マシン・トピックス」は仮想マシンの詳細を説明する。

10 章、「アプレット・インストーラ」はアプレット・インストーラの概観を与える。

11 章、「A P I 定数」はジャバ・カード 2. 1 A P I 仕様に特定されていない定数の数値を与える。

用語は、この本を使用するのに役立つ言葉とそれらの定義のリストである。

関連の書類及び刊行物

このマニュアル内ではさまざまな書類及び製品に言及する。以下の書類が入手可能である。

50

- ・ジャバ・カード 2. 1 API 草案 2 仕様、サン・マイクロシステムズ社
 - ・ジャバ・カード 2. 0 言語サブセット及び仮想マシン仕様、1997 年 10 月 13 日、改訂 1. 0 最終、サン・マイクロシステムズ社
 - ・ジャバ・カード・アプレット開発者ガイド、サン・マイクロシステムズ社
 - ・ジャバ言語使用、ジェームズ・ゴスリング、ビル・ジョイ、及びゲイ・エル・ステイール著、アディソン・ウエズレイ、1996 年、ISBN 0-201-63451-1
 - ・ジャバ仮想マシン仕様（ジャバ・シリーズ）、ティム・リインデホルム及びフランク・イエリン著、アディソン・ウエズレイ、1996 年、ISBN 0-201-63452-X
 - ・ジャバ・クラス・ライブラリ：注解参考書（ジャバ・シリーズ）パトリック
 - ・チェン及びロザンナ・リー著、アディソン・ウエズレイ、2 巻、ISBN 0201310023 及び 0201310031 10
 - ・ISO 7816 仕様 1-6 部分
 - ・支払いシステム用 EMV '96 集積回路カード仕様
- 【0077】
1. 導入
- ジャバ・カード実行時環境（JCRE）2. 1 は、ジャバ・カード仮想マシン（VM）、ジャバ・カード・アプリケーション・プログラミング・インターフェイス（API）クラス（及び業界特有拡張）及びサポート・サービスを含む。
- 【0078】
- この書類、JCRE 2. 1 仕様は、ジャバ・カード技術により要求される JCRE 機能を詳述する。ジャバ・カード技術のどんな実現もこの必要な行動と環境を与える。 20
- 【0079】
2. ジャバ・カード仮想マシンの一生
- PC 又はワークステーションで、ジャバ仮想マシンはオペレーティング・システム・プロセスとして実行する。OS プロセスが終了する時、ジャバ・アプリケーションとそれらのオブジェクトは自動的に破壊される。
- ジャバ・カード技術において、仮想マシン（VM）の実行の一生は、カードの一生である。カードに記憶された情報の大部分は電源がカードから外されても保存される。持続性メモリ技術（EEPROM など）は、スマート・カードが電源から外された時に情報の記憶を可能にする。カード上に生成された VM 及びオブジェクトは、持続性のアプリケーション情報を表すのに使用されるため、ジャバ・カード VM は永久に動作するように見える。電源が外された時、VM は単に一時的に停止する。カードが次にリセットされる時、VM は再び開始して、持続性記憶から前のオブジェクト群を回復する。 30
- 持続性の特徴は別として、ジャバ・カード仮想マシンは、ジャバ仮想マシンと同様である。
- カード初期化時は、マスキング後の時間であり、カードの個性化と発行の時間の前である。カード初期化時に、JCRE が初期化される。JCRE により生成されたフレームワーク・オブジェクトは仮想マシンの一生の間、存在する。仮想マシン及び JCRE フレームワークの実行の一生は、カードの CAD セッションに及ぶため、アプレットにより生成されたオブジェクトの一生もまた CAD セッションに及ぶ。（CAD は、カード受入装置、又はカード読取り器。カード・セッションはカードが CAD 内に挿入されて、電源が投入され、そして CAD と APDU のストリームを交換する期間である。カード・セッションはカードが CAD から取外される時に終了する。）この特性を有するオブジェクトを持続性オブジェクトと呼ぶ。 40
- JCRE 実現者は、以下の時にオブジェクトを持続性にする。
- ・Applet.register メソッドがコールされる時。JCRE はアプレット・オブジェクトのインスタンスへの参照を記憶する。JCRE 実現者は、クラス・アプレットのインスタンスが持続性であることを保証しなければならない。
 - ・オブジェクトへの参照がクラスの静的フィールド又はその他の持続性オブジェクトのフィールドに記憶される時。この要件は、JCRE 内部データ構造の完全性を保存するため 50

の必要性から派生する。

【0080】

3. ジャバ・カード・アプレットの一生

この仕様の目的のため、ジャバ・カード・アプレットの一生は、それが正しくカード・メモリにロードされて、リンクされ、又は、その他、実行のために準備された時点で開始する。（この仕様の以降において、アプレットとはジャバ・カード・プラットフォームのために書かれたアプレットを言う。）Applet.registerメソッドに登録されたアプレットは、カードの一生の間、存在する。JCREはアプレットのパブリック・メソッド、インストール、セレクト、デセレクト、及びプロセスを介してアプレットと相互作用をする。アプレットは静的インストール・メソッドを実現する。もしインストール・メソッドが実現されなければ、アプレットのオブジェクトは生成又は初期化できない。JCRE実現はアプレットのインストール、セレクト、デセレクト、及びプロセス・メソッドを以下に説明するようにコールする。

10

アプレットがスマート・カード上にインストールされる時、静的インストール・メソッドがJCREにより生成された各アプレット・インスタンスのために1度、コールされる。JCREはアプレットのコンストラクターを直接コールしてはいけない。

【0081】

3. 1メソッド・インストール

インストールがコールされる時、アプレットのオブジェクトは存在しない。アプレット内のインストール・メソッドの主な仕事は、アプレット・クラスのインスタンスを生成し、インスタンスに登録することである。アプレットがその一生の間に必要とする全ての他のオブジェクトはそれが可能な時に生成できる。その他のアプレットがCADにより選択されてアクセスされるのに必要な準備は可能な時に行なうことができる。インストール・メソッドは入力するバイト配列パラメータの内容から初期化パラメータを獲得する。

20

典型的に、アプレットは、さまざまなオブジェクトを生成し、それらを所定値で初期化し、いくつかの内部状態変数を設定し、そしてそれを選択するのに使用するためにAID（ISO7816-5に定義されるアプレット識別子）を指定するためにApplet.registerメソッドをコールする。このインストールは、Applet.registerメソッドのコールが例外なしに完了した時に成功と見なされる。もしインストール・メソッドがApplet.registerメソッドをコールしない場合、又は、もしApplet.registerメソッドがコールされる前にインストール・メソッド内から例外が投げられる場合、又は、もしApplet.registerメソッドが例外を投げる場合、にはインストールは不成功と見なされる。もしインストールが不成功ならば、JCREはそれが制御を再獲得する時に全てのクリーン・アップを実行しなければならない。すなわち、全ての持続性オブジェクトは、インストール・メソッドがコールされる前に持っていた状態に戻される。もしインストールが成功したならば、JCREはアプレットを選択可能と印を付けることができる。

30

【0082】

3. 2メソッド・セレクト

アプレットはそれらが明示的に選択されるまで、停止状態に留まる。JCREが名前データがアプレットのAIDと一致するSELECT APDUを受取る時に、選択は発生する。選択はアプレットを現在選択されたアプレットとする。

40

SELECTをコールする前、JCREは前に選択されたアプレットの選択を解除しなければならない。JCREはこれをアプレットのデセレクト・メソッドを呼出すことでアプレットに支持する。

JCREは、アプレットにそのセレクト・メソッドを呼出すことにより選択を知らせる。アプレットは、コールから誤りを返すことにより、又は例外を投げることにより、選択されることを拒否することができる。もしアプレットが真を返すと、現実のSELECT APDU命令が以後のそのプロセス・メソッドへのコールでアプレットに供給されて、アプレットがAPDU内容を検査できる。それはSELECT APDUに対してデータ（詳細はプロセス・メソッドを参照）でもって返答でき、又は、適当なSW（状態ワードの

50

返却)を有するISOExceptionを投げることでエラーのフラグを立てることができる。SW及び選択的な応答データはCADへ戻される。

Applet.selectingAppletメソッドは、セレクト・メソッド中にコールされる時に真を返答する。Applet.selectingAppletメソッドは、SELECT APDU命令を処理するためにコールされる以降のプロセス・メソッド中に真を返答し続ける。

もしアプレットが選択されることを拒否すると、JCREはCADにISO.SW_APPLET_SELECT_FAILEDのAPDU応答状態ワードを戻す。選択の失敗の際、JCRE状態はアプレットが選択されていないことを示すように設定される。

選択が成功した後、全ての以降のAPDUはプロセス・メソッドを経由して現在選択されたアプレットへ配達される。

10

【0083】

3. 3メソッド・プロセス

全てのAPDUは、JCREにより受取られ、APDUクラスのインスタンスを現在選択されたアプレットのプロセス・メソッドに送る。

注—SELECT APDUは、プロセス・メソッドのコール前に、現在選択されたアプレット内に変化を生ずるかもしれない。

正常な返答の際、JCREはアプレットにより既に送られたどんなデータにも完了応答SWとして、0x9000、を自動的に付ける。

プロセス中のどんな時にも、アプレットは適当なSWと共にISOExceptionを投げる事が可能であり、この場合、JCREはその例外を受取り、そしてSWをCADに戻す。

20

プロセス中に、もしその他の例外が投げられた場合、JCREはその例外を受取り、状態ワード、ISO7816.Sw_UNKNOWN、をCADへ戻す。

【0084】

3. 4メソッド・デセレクト

JCREが、アプレットのAIDと一致する名前のSELECT APDU命令を受取る時、JCREは現在選択されているアプレットのデセレクト・メソッドをコールする。これは他のアプレットの実行を可能にするために必要なクリーン・アップ操作をアプレットが実行することを可能にする。

Applet.selectingAppletメソッドは、デセレクト・メソッド中にコールされた時、誤りを戻す。デセレクト・メソッドにより投げられた例外はJCREにより受取られる。しかし、アプレットは選択が解除される。

30

【0085】

3. 5電源喪失及びリセット

カードがCADから取出される時又はその他の機械的又は電氣的故障が存在する場合、電源が喪失する。カードに電源が再び加えられる時及びカード・リセット（ウアーム又はコールド）時、JCREは以下を保証する。

- ・一時データはデフォルト値にリセットされる。
- ・電源が喪失した（又はリセットが発生した）時に進行中のトランザクションがあれば、中止される。

・電源が喪失した（又はリセットが発生した）時に選択されたアプレットは黙示的に選択が解除される（この場合、デセレクト・メソッドはコールされない）。

40

・もしJCREがデフォルト・アプレット選択を実現する場合は（パラグラフ5. 1参照）、デフォルト・アプレットが現在選択されたアプレットとして選択され、そして、デフォルト・アプレットのセレクト・メソッドがコールされる。それ以外は、JCREはその状態をアプレットが選択されていないことを示すものに設定する。

【0086】

4. 一時的オブジェクト

アプレットは時々、CADセッションにわたって持続する必要の無い一時的（暫定）データを含むオブジェクトを必要とする。ジャバ・カードはジャバ・キーワード遷移をサポートしない。しかし、ジャバ・カード技術はオブジェクトへの参照又はプリミティブな成分

50

を有する一時的配列を生成する方法を与える。

「一時的オブジェクト」の語は誤った名称である。これはオブジェクト自身が一時的という意味に誤解される。しかし、オブジェクトのフィールド（長さフィールドを除く）の内容のみが一時的な性格を有する。ジャバ・プログラミング言語内のその他のオブジェクトと同様に、ジャバ・カード・プラットホーム内の一時的オブジェクトは、以下から参照される限り、存在する。

- ・スタック
- ・ローカル変数
- ・クラス静的フィールド
- ・別の存在するオブジェクト内のフィールド

10

ジャバ・カード・プラットホーム内の一時的オブジェクトは以下の要求された振舞いを有する。

・一時的オブジェクトのフィールドは、ある出来事（以下を参照）の発生時に、フィールドのデフォルト値（ゼロ、誤り、又は無効）にクリアされる。

・セキュリティの理由のため、一時的オブジェクトのフィールドは「持続性メモリ技術」内には絶対に記憶されない。例として現在のスマート・カード技術を用いれば、一時的オブジェクトの内容はRAMに記憶できるが、EEPROMには決して記憶されない。この要件の目的は一時的オブジェクトがセッション・キーを記憶するのに使用できるようにするためである。

・一時的オブジェクトのフィールドへの書き込みは性能を悪くしてはいけない。（例として、現在のスマート・カード技術を取ると、一時的オブジェクトの内容はRAM内に記憶できるが、一時的でないオブジェクトの内容はEEPROM内に記憶できる。典型的に、RAM技術はEEPROMよりもずっと早い書き込みサイクルを有する。）

20

・一時的オブジェクトのファイルへの書き込みは「トランザクション」により影響を受けない。すなわち、abortTransactionは決して一時的オブジェクト内のフィールドを前の値に回復しない。

この振舞いは、一時的オブジェクトを、しばしば修正されるがCAD又は選択セッションにわたって保存する必要の無い少量の一時的アプレット・データについて、理想的にする。

。

【0087】

30

4. 1 一時的オブジェクトをクリアする出来事

持続性オブジェクトはカード・リセットをわたって持続されるべき状態を維持するために使用される。一時的オブジェクトが生成される時、そのフィールドをクリアするための1つ又は2つの出来事が指定される。クリア・オン・リセット一時的オブジェクトは、アプレット・セッションをわたって持続されるがカード・リセットをわたって持続されない状態を維持するために使用される。クリア・オン・デセレクト一時的オブジェクトは、アプレットが選択されている間は持続されるが、アプレット選択又はカード・リセットをわたっては持続しない状態を維持するために使用される。

2つのクリア出来事の詳細は以下の通りである。

・クリア・オン・リセット・オブジェクトのフィールドは、カードがリセットされる時にクリアされる。カードに電源が投入される時、これはカード・リセットを生ずる。

40

注—電源がカードから取除かれる前に一時的オブジェクトのフィールドをクリアする必要はない。しかし、このようなフィールドの内容は一旦電源が喪失すると回復できないことを保証する必要がある。

・クリア・オン・デセレクト・オブジェクトのフィールドは、アプレットの選択が解除された時には常にクリアされる。カード・リセットは黙示的に現在選択されたアプレットの選択を解除するため、クリア・オン・デセレクト・オブジェクトのフィールドはまた、クリア・オン・リセットのために指定された同じ出来事によりクリアされる。

現在選択されているアプレットは、選択命令が処理される時のみに明示的に選択が解除される。現在選択されたアプレットは選択が解除され、そして全てのクリア・オン・デセ

50

クト一時的オブジェクトのフィールドは、選択命令が以下の状態にかかわらず、クリアされる。

- ・アプレットの選択に失敗。
- ・異なるアプレットを選択。
- ・同じアプレットを再選択。

【0088】

5. 選択

カードは、A P D Uの形式でC A Dからサービスの要求を受取る。S E L E C T A P D UはJ C R Eにより現在選択されているアプレットを指定するのに使用される。一旦、選択されると、アプレットは選択が解除されるまで、全ての以降のA P D Uを受取る。

以下のいずれかが発生する時、現在選択されているアプレットは存在しない。

- ・カードがリセットされ、そしてアプレットがデフォルト・アプレットとして前もって指定されていない。
- ・アプレットの選択を試みた時、選択命令が失敗する。

【0089】

5. 1 デフォルト・アプレット

普通、アプレットは成功した選択命令を介してのみ選択される。しかし、あるスマート・カードC A Dアプリケーションは、各カード・リセット後に、黙示的に選択されるデフォルト・アプレットの存在を必要とする。その振舞いは、

1. カード・リセット（又は、リセットの形式の電源投入）後、J C R Eはその初期化を行ない、その内部状態が特定のアプレットがデフォルト・アプレットであると示しているかを検査する。もしそうならば、J C R Eはこのアプレットを現在選択されたアプレットとし、そのアプレットのセレクト・メソッドがコールされる。もしアプレットのセレクト・メソッドが例外を投げ又は誤りを返すと、J C R Eはその状態をアプレットが選択されないことを示す状態に設定する。（デフォルト・アプレット選択中にはS E L E C T A P D Uが無い場合、アプレットのプロセス・メソッドはコールされない。）デフォルト・アプレットがカード・リセットにおいて選択される時、そのプロセス・メソッドがコールされる必要はない。

2. J C R EはA T Rが送られることを保証し、そしてカードは現在A P D U命令を受取るための用意ができています。

もしデフォルト・アプレットが成功的に選択されると、A P D U命令はこのアプレットに直接的に送られることができる。もしデフォルト・アプレットが選択されなければ、選択命令のみが処理できる。

デフォルト・アプレットを指定するための機構はジャバ・カードA P I 2. 1には定義されていない。それはJ C R E実現の詳細であり、個々のJ C R E実現者に任されている。

【0090】

5. 2 選択命令処理

S E L E C T A P D U命令はアプレットを選択するのに使用される。その振舞いは、

1. アプレットがもしあれば活動的かどうかにかかわらず、J C R EによりS E L E C T A P D Uは常に処理される。

2. J C R Eは一致したA I Dについて内部テーブルを探す。J C R Eは選択命令中に完全なA I Dがある場合、アプレットの選択を支援する。

J C R E実現者は、他の選択基準を支援するために彼等のJ C R Eを増強することは自由である。この一例は、I S O 7 8 1 6 - 1 4に説明されるように部分的なA I D一致を介しての選択である。特別な要件は以下の通りである。

注一星印はI S O 7 8 1 6内の2進値ビット番号を示す。最上位ビット=b 8、最下位ビット=b 1である。

a) アプレット選択命令は、CLA=0x00, INS=0xA4、を使用する。

b) アプレット選択命令は、「D F名による選択」を使用する。従って、P1=0x04である。

10

20

30

40

50

c) P1の他の値はアプレット選択を示唆しない。A P D Uは現在選択されたアプレットにより処理される。

d) J C R Eは、正確なD F名 (A I D) 選択、すなわち、P2=%b0000xx00(b4,b3*は無視)、を支援する。

e) 全ての他の部分D F名選択オプション(b2,b1*)は、J C R E実現に依存する。

f) 全てのファイル制御情報オプション・コード(b4,b3*)は、J C R Eにより支援され、そしてアプレットにより解釈されそして処理される。

3. もしA I D一致が見つからなければ、

a. もし現在選択されたアプレットが存在しなければ、J C R Eは選択命令に状態コード0x6999(SW_APPLET_SELECT_FAILED)でもって返答する。

10

b. それ以外は、選択命令は、現在選択されたアプレットのプロセス・メソッドに送られる。この点で、アプレットのコンテキストへのコンテキスト切換えが発生する。(アプレット・コンテキストはパラグラフ6. 1. 1. で定義される。) アプレットは、それら自身の内部選択処理のためにS E L E C T A P D U命令を使用できる。

4. もし一致したA I Dが見つれば、J C R Eは新しいアプレットを選択する準備をする。もし現在選択されたアプレットが存在すれば、そのデセレクト・メソッドへのコールを介して選択解除される。この時点で選択解除されたアプレットのコンテキストへのコンテキスト切換えが生ずる。J C R Eコンテキストはデセレクトからの出口において回復される。

5. J C R Eは、新たに現在選択されたアプレットを設定する。新しいアプレットは、そのセレクト・メソッドへのコールを介して選択される。そして新しいアプレットのコンテキストへのコンテキスト切換えが発生する。

20

a. もしアプレットのセレクト・メソッドが例外を投げる又は誤りを戻すと、J C R E状態はアプレットが選択されないように設定される。J C R Eは選択命令に対して状態コード0x6999(SW_APPLET_SELECT_FAILED)でもって返答する。

b. 新たに現在の選択されたアプレットのプロセス・メソッドは、入力パラメータとしてS E L E C T A P D Uと共にコールされる。アプレットのコンテキストへのコンテキスト切換えが発生する。

注ーもし一致するA I Dが存在しなければ、選択命令は通常のアプレットA P D U命令として処理するために現在選択されたアプレット(もし有れば)に送られる。

30

もし、一致するA I Dが存在し、そして選択命令が失敗すると、J C R Eは常にアプレットが選択されない状態にはいる。

もし、一致するA I Dが現在選択されたアプレットと同じならば、J C R Eはアプレットの選択を解除するプロセスを経過して、そしてそれを選択する。再選択は失敗することがあり、カードをアプレットが選択されない状態にする。

【0091】

5. 3非選択命令処理

非選択A P D Uが受取られ、そして現在選択されたアプレットが存在しない時、J C R EはA P D Uに状態コード0x6999(SW_APPLET_SELECT_FAILED)でもって返答する。

非選択A P D Uが受取られ、そして現在選択されたアプレットが存在する時、J C R Eは現在選択されたアプレットのプロセス・メソッドを呼出して、A P D Uをパラメータとして送る。これは、J C R Eコンテキストから現在選択されたアプレットのコンテキストへのコンテキスト切換えを生ずる。プロセス・メソッドが存在する時、V MはJ C R Eコンテキストへ戻される。J C R Eは返答をA P D Uへ送る。そして次の命令A P D Uを待つ。

40

【0092】

6. アプレット隔離及びオブジェクト共有

J C R Eのどんな実現もコンテキスト及びアプレットの隔離を支援する。隔離とは、他のアプレットが明示的にアクセスのためのインターフェイスを与えなければ、アプレットは別のコンテキスト内のアプレットのオブジェクト又はフィールドにアクセスすることがで

50

きないことを意味する。

【0093】

6. 1 アプレット・ファイヤウォール

ジャバ・カード技術内のアプレット・ファイヤウォールは実行時実施保護であり、ジャバ技術保護とは別である。ジャバ言語保護はジャバ・カード・アプレットへ適用される。ジャバ言語は強いタイピングと保護属性を実施することを保証する。

アプレット・ファイヤウォールは常にジャバ・カードVM内で実施される。これらは、VMを実行時に自動的に追加のセキュリティ検査を実行することを可能にする。

【0094】

6. 1.1 コンテキスト及びコンテキスト切換え

ファイヤウォールは本質的にジャバ・カード・プラットフォームのオブジェクト・システムを、コンテキストと呼ばれる分離した保護されたオブジェクト空間に区分する。ファイヤウォールは、1つのコンテキストと別の間の境界である。JCREはカード上にインストールされている各アプレットに対するアプレット・コンテキストを割当てそして管理する（しかし、グループ・コンテキストの議論について下のパラグラフ6. 1. 1. 2を参照）。

これに加え、JCREはそれ自身のJCREコンテキストを維持する。このコンテキストはアプレット・コンテキストと大変似ている。しかし、それはアプレット・コンテキストには否定された操作を実行することができるように特別なシステム特権を有する。

どの時点においても、VM内には1つのみの活動的なコンテキスト存在する。（これが現在活動的なコンテキストと呼ばれる。）オブジェクトにアクセスする全てのバイトコードは、アクセスが許可されるかどうかを決定するために、現在活動的なコンテキストに対して実行時に検査される。アクセスが不許可の時、`java.lang.SecurityException`が投げられる。

パラグラフ6. 2. 8に記述するように、呼出しタイプ・バイトコードの実行中に、ある良く定義された条件が満足される時、VMはコンテキスト切換えを実行する。前のコンテキストは、内部VMスタックに押し込められて、新しいコンテキストが現在活動的なコンテキストとなり、呼出されたメソッドがこの新しいコンテキスト内で実行する。メソッドから出る時、VMはコンテキスト切換えの回復を実行する。元のコンテキスト（メソッドをコールした）はスタックから取出されて、現在活動的なコンテキストとして回復される。コンテキスト切換えはネストできる。最大の深さは、利用可能なVMスタック空間の量に依存する。

ジャバ・カード技術内での大部分のメソッドの呼出しは、コンテキスト切換えを生じない。コンテキスト切換えは、あるメソッドからの戻りと呼出し中及びそれらのメソッドからの例外出口中のみを生ずる（6. 2. 8参照）。

コンテキスト切換えメソッド呼出し中、現在活動的なコンテキストを示すデータの追加の部分が戻りスタック内に押し込まれる。メソッドが出る時にこのコンテキストが回復される。

この章の以下の部分においてコンテキスト及びコンテキスト切換えのさらなる詳細が与えられる。

【0095】

6. 1. 1. 1 グループ・コンテキスト

普通、ジャバ・カード・アプレットの各インスタンスは別のコンテキストを定義する。しかし、ジャバ・カード2. 1技術では、グループ・コンテキストの概念が導入される。もし2以上のアプレットが、1つのジャバ・パッケージ内に含まれる時、それらは同じコンテキストを共有する。これに加え、同じアプレット・クラスの全てのインスタンスは同じコンテキストを共有する。換言すれば、グループ・コンテキスト内の2つのアプレット・インスタンス間にファイヤウォールは存在しない。

節6. 1. 1内の上記コンテキスト及びコンテキスト切換えの説明は、各アプレット・インスタンスが別のコンテキストと関連していると仮定している。ジャバ・カード2. 1技

10

20

30

40

50

術では、ファイヤウォールを実施するためにコンテキストが比較される。そしてインスタンス A I D がスタック内に押し込まれる。これに加えて、これはコンテキスト切換えの時だけではなく、1つのアプレット・インスタンスにより所有されたオブジェクトから同じパッケージ内の別のインスタンスにより所有されたオブジェクトへの制御切換えの時にも生ずる。

【0096】

6. 1. 2 オブジェクト所有

新しいオブジェクトが生成される時、それは現在活動的なコンテキストと関連付けられる。しかし、オブジェクトは、オブジェクトが具体化される時に現在活動的なコンテキスト内のアプレット・インスタンスにより所有される。オブジェクトはアプレット・インスタンス又は J C R E により所有される。

10

【0097】

6. 1. 3 オブジェクト・アクセス

一般に、オブジェクトはその所有するコンテキストによりのみアクセスできる。すなわち、所有コンテキストが現在活動的なコンテキストである時。ファイヤウォールが異なるコンテキスト内の別のアプレットによりオブジェクトがアクセスされることを防ぐ。

実現の観点では、オブジェクトがアクセスされる時は何時でも、オブジェクトの所有コンテキストが現在活動的なコンテキストと比較される。もしこれらが一致しなければ、アクセスは実行されず、SecurityException が投げられる。

オブジェクトの参照を使用して、以下のバイトコードの1つが実行される時、オブジェクトがアクセスされる。

20

Getfield, putfield, invokevirtual, invokeinterface, athrow,

<T>aload, <T>astore, arraylength, checkcast, instanceof,

<T> は、baload, sastore等、さまざまなタイプのバイトコード配列を言う。

このリストは、getfield_b, sgetfield_s_this,等、ジャバ・カード V M 内で実現されるこれらのバイトコードの特別な又は最適化された形を含む。

【0098】

6. 1. 4 ファイヤウォール保護

ジャバ・カード・ファイヤウォールは、最もしばしば予期されるセキュリティ問題、大切なデータを別のアプレットに「漏らす」ような開発者の誤り及び設計の見過ごし、に対して保護を与える。アプレットは公にアクセス可能な場所からオブジェクトの参照を得ることができるかもしれないが、しかし、オブジェクトが異なるアプレットに所有されていると、ファイヤウォールはセキュリティを保証する。

30

ファイヤウォールはまた、間違っただけのコードに対して保護を与える。もしカードに誤ったコードがロードされると、ファイヤウォールはこのコードからオブジェクトがアクセスされることを保護する。

ジャバ・カード A P I 2. 1 は、これらの特徴がアプレット開発者には透明ではない方法で支援されるため、ファイヤウォール及びコンテキストの基本的な最小保護要件を説明する。開発者は、A P I、ファイヤウォールに関する例外、及び、オブジェクトの振舞いについて知らなければならない。

40

J C R E 実現は、これらの機構がアプレットに透明で且つ V M の外部に見える操作を変更しない限り、これらのアプレット・ファイヤウォールを越えて、追加のセキュリティ機構を実現することは自由である。

【0099】

6. 1. 5 静的フィールド及びメソッド

クラスはコンテキストにより所有されないことに注意すべきである。クラス静的フィールドがアクセスされる時、実行される実行時コンテキスト検査はない。静的メソッドが呼出される時、コンテキスト切換えはない。(同様に、invokespecialはコンテキスト切換えを生じない)

パブリック静的フィールド及びパブリック静的メソッドは、どんなコンテキストからもア

50

クセス可能である。静的メソッドはそれらと呼出したのと同じコンテキスト内で実行される。

静的フィールド内で参照されるオブジェクトは普通のオブジェクトである。これらはこれらを生成したものより所有され、標準のファイヤウォール・アクセス規則が適用される。もしそれらが複数のアプレット・コンテキストに横断して共有される必要があるならば、これらは共有可能インターフェイス・オブジェクト（SIO）である必要がある。（以下のパラグラフ6. 2. 4参照）

もちろん、通常のジャバ技術保護が静的フィールド及びメソッドに対して実施できる。これに加え、アプレットがインストールされる時、インストーラは外部の静的フィールド又はメソッドへのリンクの各試みが許可されることを確認する。インストレーションとリン

10

【0100】

6. 1. 5. 1 選択的静的アクセス検査

JCREは、確認者により実施される制約と冗長なオプションな実行時検査を実行できる。ジャバ・カードVMは、コードが別のクラスのプライベートなメソッドの呼出しなど、基本的な言語制限に違反する時、検出して、報告し、又は違反に対処する。

【0101】

6. 2 コンテキストを横断するオブジェクトへのアクセス

アプレットが互いに相互作用し、又は、JCREと相互作用することを可能にするため、1つのコンテキストが別のコンテキストに属するオブジェクトにアクセスすることができ

20

る、良く定義され且つ安全な機構が提供される。これらの機構は、ジャバ・カードAPI 2. 1において与えられ、そして以下の節で説明される。

- ・ JCRE入口点オブジェクト
- ・ グローバル配列
- ・ JCRE特権
- ・ 共有可能インターフェイス

【0102】

6. 2. 1 JCRE入口点オブジェクト

安全なコンピュータ・システムは、非特権ユーザ・プロセス（資源の一部に限定された）が、特権的な「システム」ルーチンにより実行されるシステム・サービスを要求するための方法を持たなければならない。

30

ジャバ・カードAPI 2. 1では、これはJCRE入口点オブジェクトを使用して達成される。これらはJCREコンテキストにより所有されるオブジェクトであるが、しかしそれらは入口点メソッドを含むとしてフラグが立てられている。

ファイヤウォールは、これらのオブジェクトをアプレットによるアクセスから保護する。入口点指定はこれらのオブジェクトのメソッドがどんなコンテキストからも呼出されることを可能にする。それが発生する時、JCREコンテキストへのコンテキスト切り換えが実行される。これらのメソッドは、アプレットが特権JCREシステム・サービスを要求するゲートである。

40

JCRE入口点オブジェクトには2つの分類がある。

- ・ 一時的JCRE入口点オブジェクト

全てのJCRE入口点オブジェクトと同様に、一時的JCRE入口点オブジェクトのメソッドは、どんなアプレットのコンテキストから呼出されることができる。しかし、これらのオブジェクトへの参照は、クラス変数、インスタンス変数、又は配列要素内には記憶することができない。JCREは、これらのオブジェクトへの参照の記憶の試みを、認証されない再使用を防ぐために、ファイヤウォール機能の一部として検出して制限する。

APDUオブジェクト及び全てのJCRE所有例外オブジェクトは、一時的JCRE入口点オブジェクトの例である。

- ・ 永久JCRE入口点オブジェクト

50

全ての J C R E 入口点オブジェクトと同様に、永久 J C R E 入口点オブジェクトのメソッドは、どんなアプレットのコンテキストから呼出されることができる。これに加えて、これらのオブジェクトへの参照は記憶でき、自由に再使用できる。

J C R E 所有 A I D インスタンスは、永久 J C R E 入口点ポイントの例である。

J C R E は、以下の責任を有する。

- ・どんな特権的サービスがアプレットに提供できるかを決定する。
- ・それらのサービスへの入口点メソッドを含むクラスを定義する。
- ・それらのクラスの 1 又は複数のオブジェクト・インスタンスを生成する。
- ・それらのインスタンスを J C R E 入口点オブジェクトと指定する。
- ・ J C R E 入口点オブジェクトを一時的又は永久と指定する。
- ・必要に応じてこれらのオブジェクトへの参照をアプレットに利用可能にする。

10

注—これらのオブジェクトのメソッドのみがファイヤウォールを通じてアクセス可能にする。これらのオブジェクトのフィールドはファイヤウォールにより保護されていて、J C R E コンテキストのみによりアクセス可能である。

J C R E 自身のみが入口点オブジェクト及びそれらが一時的又は永久かを指定できる。J C R E 実現者は、J C R E 入口点オブジェクトが指定されそしてそれらがどのように一時的又は永久的になるかの機構の実現に責任を有する。

【0103】

6. 2. 2 グローバル配列

いくつかのオブジェクトのグローバル性質はそれらがどんなアプレット・コンテキストからもアクセス可能であることを要求する。ファイヤウォールは通常、これらオブジェクトが柔軟な態様で使用されることを防ぐ。ジャバ・カード V M はオブジェクトがグローバルとして指定されることを可能にする。

20

全てのグローバル配列は一時的グローバル配列オブジェクトである。これらオブジェクトは J C R E コンテキストにより所有されるが、どんなアプレット・コンテキストからもアクセスできる。しかし、これらのオブジェクトへの参照はクラス変数、インスタンス変数又は配列要素内には記憶できない。J C R E はこれらのオブジェクトへの参照の記憶の試みを、認証されない再使用を防ぐためにファイヤウォール機能の一部として検出して制限する。

追加のセキュリティのため、配列のみがグローバルと指定でき、そして J C R E 自身のみがグローバル配列を指定できる。アプレットがそれらを生成できないため、A P I メソッドは定義されない。J C R E 実現者はグローバル配列が指定される機構を実現する責任がある。

30

この仕様の刊行時に、ジャバ・カード A P I 2. 1 に要求されるグローバル配列は、A P D U バッファ及びアプレットのインストール・メソッドへのバイト配列入力パラメータ (b 配列) のみである。

注—そのグローバル地位のため、アプレットが選択される時は常に、J C R E が新しい A P D U 命令を受取る前に、A P D U バッファがゼロにクリアされることを A P I は指定する。これは、アプレットの潜在的な重要なデータがグローバル A P D U バッファを経由して他のアプレットへ「漏れる」ことを防止する。A P D U バッファは、共有インターフェイス・オブジェクト・コンテキストからアクセスでき、アプレット・コンテキストを横断してデータを送るのに適している。アプレットは、A P D U バッファからアクセスされる秘密データを保護する責任がある。

40

【0104】

6. 2. 3 J C R E 特権

「システム」コンテキストであるため、J C R E コンテキストは特別な特権を有する。それはカード上のどんなオブジェクトのメソッドも呼出することができる。例えば、オブジェクト X がアプレット A により所有されていると仮定すると、普通、コンテキスト A のみがフィールド及び X のメソッドをアクセスできる。しかし、J C R E コンテキストは X のどんなメソッドも呼出することができる。このような呼出し中、コンテキスト切換えが、J C

50

R E コンテキストから X を所有する アプレット・コンテキストへ生ずる。

注— J C R E は、X のメソッド及びフィールドの両方をアクセスできる。メソッドのアクセスは J C R E が アプレット・コンテキストに入るための機構である。J C R E はファイヤウォールを介してどんなメソッドも呼出すことができるが、アプレット・クラスで定義された、select, process, deselect, getShareableInterfaceObject メソッド（6. 2. 7. 1 参照）のみを呼出すべきである。

J C R E コンテキストは、V M がカード・リセット後に実行を開始する時の現在活動的なコンテキストである。J C R E コンテキストは「ルート」コンテキストであり、常に現在活動的なコンテキスト又はスタックに記憶された底コンテキストのいずれかである。

【0105】

6. 2. 4 共有可能インターフェイス

共有可能インターフェイスは、アプレット相互作用を可能にするためのジャバ・カード A P I 2. 1 の新しい特徴である。共有可能インターフェイスは、1 組の共有されたインターフェイス・メソッドを定義する。これらのインターフェイス・メソッドは、もしそれらを実現するオブジェクトが別のアプレット・コンテキストにより所有されていても、1 つのアプレット・コンテキストから呼出すことができる。

この仕様では、共有可能インターフェイスを実現するクラスのオブジェクト・インターフェイスは、共有可能インターフェイス・オブジェクト（S I O）と呼ばれる。

所有コンテキストには、S I O はそのフィールド及びメソッドがアクセスできる普通のオブジェクトである。その他のコンテキストには、S I O は共有可能なインターフェイスのインスタンスであり、共有可能インターフェイス内で定義されたメソッドのみがアクセス可能である。S I O のその他の全てのフィールド及びメソッドはファイヤウォールにより保護される。

共有可能インターフェイスは、アプレット間通信のための安全な機構を以下の様に与える。

1. オブジェクトを別のアプレットに利用可能にするため、アプレット A は最初に共有可能インターフェイス S I を定義する。共有可能インターフェイスはインターフェイス、javacard.framework.Shareable、を拡張する。共有可能インターフェイス S I に定義されたメソッドは、アプレット A が他のアプレットにアクセス可能にしたサービスを表す。

2. そして、アプレット A は共有可能インターフェイス S I を実現するクラス C を定義する。C は S I 内に定義されたメソッドを実現する。C は他のメソッド及びフィールドも定義してもよいが、これらはアプレット・ファイヤウォールで保護されている。S I 内で定義されているメソッドのみが他のアプレットにとりアクセス可能である。

3. アプレット A は、クラス C のオブジェクト・インスタンス O を生成する。O はアプレット A に属し、ファイヤウォールは A が O のどんなフィールド及びメソッドもアクセスすることを可能にする。

4. アプレット A のオブジェクト O にアクセスするために、アプレット B はタイプ S I のオブジェクト参照 S I O を生成する。

5. アプレット B は、アプレット A から共有されたインターフェイス・オブジェクト参照を要求するために特別なメソッド（パラグラフ 6. 2. 7. 2 に記載された J C System. getAppletShareableInterfaceObject）を呼出す。

6. アプレット A は、要求及び Applet. getShareableInterfaceObject を経由して要求者（B）の A I D を受取り、そしてそれがアプレット B とオブジェクト O を共有するかどうかを決定する。

7. もしアプレット A が、アプレット B と共有を同意すれば、A は O への参照と共に要求に応答する。この参照は O のメソッド又はフィールドのいずれも見えないようにするために、Shareable をタイプするためのキャストである。

8. アプレット B はアプレット A からオブジェクト参照を受取り、それを S I をタイプするためにキャストし、それをオブジェクト参照 S I O 内に記憶する。S I O は実際に A のオブジェクト O を参照するが、S I O はタイプ S I である。S I で定義された共有可能イ

10

20

30

40

50

ンターフェイス・メソッドのみがBに見える。ファイヤウォールはOの他のフィールド及びメソッドがBによりアクセスされることを防止する。

9. アプレットBは、S I Oの共有可能なインターフェイス・メソッドの1つを呼出すことにより、アプレットAからのサービスを要求できる。呼出し中、ジャバ・カードVMはコンテキスト切換えを実行する。元の現在活動的なコンテキスト(B)はスタック上に記憶され、そして実際のオブジェクト(O)の所有者(A)のコンテキストが新しく現在活動的なコンテキストとなる。共有可能インターフェイス・メソッド(S I メソッド)のAの実現がAのコンテキスト中で実行される。

10. S I メソッドは、JCSys_{tem}.getPreviousContextAIDメソッドを介してそのクライアント(B)のA I Dを見付けることができる。これはパラグラフ6. 32. 5内に記載されている。このメソッドはアプレットBのためにサービスを実行するかどうかを決定する。

10

11. コンテキスト切換えのため、ファイヤウォールは、S I メソッドがオブジェクトOの全てのメソッド及びフィールド及びAにより所有されるその他のオブジェクトにアクセスすることを可能にする。同時に、ファイヤウォールはメソッドがBにより所有された非共有オブジェクトにアクセスすることを防止する。

12. S I メソッドは、Bにより送られたパラメータにアクセスでき、そしてBに戻り値を与えることができる。

13. 戻る際に、ジャバ・カードVMはコンテキスト切換えの回復を実行する。元の現在活動的なコンテキスト(B)がスタックから押出されて、再び、現在のコンテキストとなる。

20

14. コンテキスト切換えのため、ファイヤウォールは再び、Bがそのオブジェクトのいずれにもアクセスすることを可能にし、そしてBがA所有の非共有オブジェクトにアクセスすることを防止する。

【0106】

6. 2. 5 前のコンテキストの決定

アプレットが、JCSys_{tem}.getPreviousContextAIDをコールする時、J C R Eは最後のコンテキスト切換えの時に活動的なアプレット・インスタンスのインスタンスA I Dに戻す。

【0107】

6. 2. 5. 1 J C R E コンテキスト

30

J C R E コンテキストはA I Dを持たない。アプレット・コンテキストがJ C R Eから直接に入った時、もしアプレットが、getPreviousContextAIDをコールすれば、このメソッドは無効を返答する。

もしアプレットが、アプレット自身から又は共有可能インターフェイスを介してアクセスされている時に外部アプレットからアクセスされているメソッドから、getPreviousContextAIDをコールすれば、それは呼出し者A I D認証を実行する前に、無効返答を検査する。

【0108】

6. 2. 6 共有可能インターフェイス詳細

共有可能インターフェイスは、単にタグging・インターフェイス、javacard.framework.Shareable、を拡張(直接的又は間接的のいずれかで)したものである。この共有可能インターフェイスはR M I フアシリテイにより使用された遠隔インターフェイスの概念的に類似していて、インターフェイス・メソッドへのコールは、ローカル/遠隔境界を横断して生ずる。

40

【0109】

6. 2. 6. 1 ジャバ・カード共有可能インターフェイス

共有可能タグging・インターフェイスを拡張したインターフェイスはこの特別な性質を有する。インターフェイス・メソッドのコールはコンテキスト切換えによりジャバ・カードのアプレットのファイヤウォール境界を横断して生ずる。

共有可能インターフェイスは全ての共有されたオブジェクトを識別する役割を有する。ア

50

プレット・ファイヤウォールを介して共有される必要のあるオブジェクトはこのインターフェイスを直接的に又は間接的に実現する。共有可能なインターフェイスに指定されたこれらの方法のみがファイヤウォールを介して利用可能である。

実現クラスはどんな数の共有可能なインターフェイスも実現でき、その他の共有可能な実現クラスに拡張できる。

ジャバ・プラットフォーム・インターフェイスと同じく、共有可能インターフェイスは単に1組のサービス・メソッドを定義する。サービス提供者クラスが共有可能インターフェイスを「実現」することを宣言し、インターフェイスのサービス・メソッドの各々の実現を提供する。サービス・クライアント・クラスが、オブジェクト参照を得て、もし必要ならば、共有可能インターフェイス・タイプにそれをキャストし、そしてインターフェイスのサービス・メソッドを呼び出すことにより、サービスにアクセスする。

10

ジャバ・カード技術内の共有可能インターフェイスは以下の性質を有する。

- ・共有可能インターフェイス内のメソッドが呼出された時、コンテキスト切換えがオブジェクト所有者のコンテキストへ生ずる。
- ・メソッドが出る時、コールしたもののコンテキストが回復される。
- ・例外が投げられた時に発生するスタック・フレームの巻き戻しの際に、現在活動的なコンテキストが正しく回復されるように、例外処理が增強される。

【0110】

6. 2. 7 共有可能インターフェイス・オブジェクトの獲得

アプレット間通信は、クライアント・アプレットがサーバー・アプレットに属するSIOの共有可能インターフェイス・メソッドを呼出す時に達成される。これが動作するために、クライアント・アプレットが最初にサーバー・アプレットからSIOを獲得するための方法がなければならない。JCREはこれを可能にする機構を与える。アプレット・クラス及びJCSysmクラスはクライアントがサーバーからサービスを要求することを可能にするメソッドを与える。

20

【0111】

6. 2. 7. 1 メソッド・アプレット.getShareableInterfaceObject

このメソッドは、サーバー・アプレット・インスタンスにより実現される。それは別のアプレットに属するオブジェクトを使用することを要求するクライアント・アプレットとオブジェクトを共有可能にするサーバー・アプレットとの間の調停をするため、JCREによりコールされなければならない。

30

デフォルトの振舞いは無効を返答する。これはアプレットはアプレット間通信に参加しないことを示す。

別のアプレットから呼出されることを意図されたサーバー・アプレットはこのメソッドを覆す必要が有る。このメソッドはクライアントAIDとパラメータを検査する。もしクライアントAIDが期待されたAIDの1つでなければ、メソッドは無効を返答する。同様に、もしパラメータが認識されなければ、又は、もしそれがクライアントAIDに許可されなければ、メソッドはまた無効を返す。それら以外は、アプレットはクライアントが要求した共有可能インターフェイス・タイプのSIOを戻す。

サーバー・アプレットは、同じSIOを全てのクライアントに返答する必要は無い。サーバーは異なる目的のために共有されたインターフェイスの複数のタイプを支援でき、クライアントへの種類のSIOを戻すかを決定するためにクライアントAIDとパラメータを使用する。

40

【0112】

6. 2. 7. 2 メソッドJCSysm.getAppletShareableInterfaceObject

JCSysmクラスはメソッドgetAppletShareableInterfaceObjectを含む。これはサーバー・アプレットと通信するためにクライアント・アプレットにより呼出される。

JCREは以下の様に振舞うためにこのメソッドを実現する。

1. JCREはその内部アプレット・テーブル内でサーバーAIDを持ったものを探す。見つからなければ、無効が戻される。

50

2. J C R Eは、このアプレットのgetShareableInterfaceObjectメソッドを呼出し、呼出したもののクライアントA I Dとパラメータを送る。

3. コンテキスト切換えがサーバー・アプレットへ生ずる。そして前記したようにgetShareableInterfaceObjectの実現が進行する。サーバー・アプレットはS I O（又は無効）を戻す。

4. GetAppletShareableInterfaceObjectは同じS I O（又は無効）をそのコールしたものに戻す。

増強されたセキュリティのため、実現はクライアントが以下の状態のどれが無効値を戻したかを知ることが不可能にする。

- ・サーバーA I Dが見つからない。
- ・サーバー・アプレットはアプレット間通信に参加していない。
- ・サーバー・アプレットはクライアントA I D又はパラメータを認識しない。
- ・サーバー・アプレットはこのクライアントと通信しない。
- ・サーバー・アプレットはパラメータにより指定されたようにこのクライアントと通信しない。

10

【0113】

6. 2. 8 クラス及びオブジェクト・アクセスの振舞い

静的クラス・フィールドは、以下の1つのジャバ・バイトコードが実行される時にアクセスされる。

getstatic, putstatic

20

オブジェクトは、以下の1つのジャバ・バイトコードがオブジェクト参照を使用して実行される時にアクセスされる。

Getfield, putfield, invokevirtual, invokeinterface, athrow,

<T>aload, <T>astore, arraylength, checkcast, instanceof,

<T> は、baload, sastore等、さまざまなタイプのバイトコード配列を言う。

このリストは、getfield_b, sgetfield_s_this,等、ジャバ・カードVM内で実現されるこれらのバイトコードの特別な又は最適化された形を含む。

ジャバVMにより指定されたバイトコードの仕事を実行する前に、ジャバ・カードVMは参照されたオブジェクト上にアクセス検査を実行する。もしアクセスが否定されると、SecurityExceptionが投げられる。

30

ジャバ・カードVMにより実行されるアクセス検査は参照されたオブジェクト、バイトコード、及び現在活動的なコンテキストのタイプと所有者に依存する。それらは以降に説明される。

【0114】

6. 2. 8. 1 静的クラス・フィールドへのアクセス

バイトコード：

getstatic, putstatic

- ・もしJ C R Eが現在活動的なコンテキストであれば、アクセスが許可される。
- ・それ以外で、もしバイトコードがputstaticで、記憶されているフィールドが参照タイプで、記憶されている参照が一時J C R E入口点オブジェクト又はグローバル配列への参照であれば、アクセスは否定される。
- ・それ以外は、アクセスは許可される。

40

【0115】

6. 2. 8. 2 配列オブジェクトへのアクセス

バイトコード：

<T>aload, <T>astore, arraylength, checkcast, instanceof

- ・もしJ C R Eが現在活動的なコンテキストであれば、アクセスが許可される。
- ・それ以外は、もしバイトコードがastoreで、記憶されている要素が参照タイプであり、記憶されている参照が一時J C R E入口点オブジェクト又はグローバル配列への参照であれば、アクセスは否定される。

50

・それ以外で、もし配列が現在活動的コンテキストにより所有されていれば、アクセスは許可される。

・それ以外で、配列がグローバルと指定されれば、アクセスは許可される。

・それ以外は、アクセスは否定される。

【0116】

6. 2. 8. 3 クラス・インスタンス・オブジェクト・フィールドへのアクセス

バイトコード：

getfield, putfield

・もし J C R E が現在活動的なコンテキストであれば、アクセスが許可される。

・それ以外で、もしバイトコードが putfield で、記憶されているフィールドが参照タイプで、記憶されている参照が一時 J C R E 入口点オブジェクト又はグローバル配列への参照であれば、アクセスは否定される。

・それ以外で、もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。

・それ以外は、アクセスは否定される。

【0117】

6. 2. 8. 4 クラス・インスタンス・オブジェクト・メソッドへのアクセス

バイトコード：

invokevirtual

・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。コンテキストはオブジェクトの所有者のコンテキストへ切り換えられる。

・それ以外で、もしオブジェクトが J C R E 入口点オブジェクトと指定されればアクセスは許可される。コンテキストはオブジェクトの所有者のコンテキスト（J C R E の）へ切り換えられる。

・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスが許可される。コンテキストはオブジェクトの所有者のコンテキストへ切り換えられる。

・それ以外は、アクセスが否定される。

【0118】

6. 2. 8. 5 標準インターフェイス・メソッドへのアクセス

バイトコード：

invokeinterface

・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。

・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスが許可される。コンテキストはオブジェクトの所有者のコンテキストへ切り換えられる。

・それ以外は、アクセスは拒否される。

【0119】

6. 2. 8. 6 共有可能インターフェイス・メソッドへのアクセス

バイトコード：

invokeinterface

・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。

・それ以外で、もしオブジェクトのクラスが共有可能インターフェイスを実現すれば、そしてもし呼出されるインターフェイスが共有可能インターフェイスを拡張すれば、アクセスは許可される。コンテキストはオブジェクトの所有者のコンテキストへ切り換えられる。

・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスが許可される。コンテキストはオブジェクトの所有者のコンテキストへ切り換えられる。

・それ以外は、アクセスは拒否される。

【0120】

6. 2. 8. 7 例外オブジェクトを投げる

バイトコード：

athrow

- ・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。
- ・それ以外で、もしオブジェクトが J C R E 入口点オブジェクトと指定されれば、アクセスは許可される。
- ・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスは許可される。
- ・それ以外は、アクセスは否定される。

10

【0 1 2 1】

6. 2. 8. 8 クラス・インスタンス・オブジェクトへのアクセス

バイトコード

checkcast, instanceof

- ・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。
- ・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスは許可される。
- ・それ以外で、もしオブジェクトが J C R E 入口点オブジェクトと指定されれば、アクセスは許可される。
- ・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスは許可される。
- ・それ以外は、アクセスは否定される。

20

【0 1 2 2】

6. 2. 8. 9 標準インターフェイスへのアクセス

バイトコード

checkcast, instanceof

- ・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。
- ・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスは許可される。
- ・それ以外は、アクセスは否定される。

30

【0 1 2 3】

6. 2. 8. 10 共有可能インターフェイスへのアクセス

バイトコード

checkcast, instanceof

- ・もしオブジェクトが現在活動的なコンテキストに所有されれば、アクセスは許可される。
- ・それ以外で、もしオブジェクトのクラスが共有可能インターフェイスを実現し、そしてもしオブジェクトが共有可能インターフェイスを拡張するインターフェイスにキャスト (checkcast) され又はそのインスタンス (instanceof) であれば、アクセスは許可される。
- ・それ以外で、もし J C R E が現在活動的なコンテキストであれば、アクセスは許可される。
- ・それ以外は、アクセスは否定される。

40

【0 1 2 4】

6. 3 一時的オブジェクト及びアプレット・コンテキスト

クリア・オン・リセット・タイプの一時的オブジェクトは、現在活動的なアプレット・コンテキストがオブジェクトの所有者（オブジェクトが生成された時に現在活動的なアプレット・コンテキスト）と同じ時のみにそれらがアクセスできるという点で、持続性オブジェクトと似た振舞いをする。

50

クリア・オン・デセレクト・タイプの一時的オブジェクトは、現在活動的なアプレット・コンテキストが現在選択されたアプレット・コンテキストである時のみに生成又はアクセスできる。もしどんなmakeTransientfactoryメソッドが、現在活動的なアプレット・コンテキストが現在選択されたアプレット・コンテキストでない時に、クリア・オン・デセレクト・タイプの一時的オブジェクトを生成するためにコールされると、メソッドは理由コード、ILLEGAL_TRANSIENT、と共に、SystemException、を投げる。もしクリア・オン・デセレクト・タイプの一時的オブジェクトへのアクセスが、現在活動的なアプレット・コンテキストが現在選択されたアプレット・コンテキストでない時に試みられた場合、J C R EはSecurityExceptionを投げる。

同じパッケージの部分のアプレットは同じグループ・コンテキストを共有する。パッケージからの各アプレット・インスタンスは、同じパッケージからの全ての他のインスタンスと共に全てのそのオブジェクト・インスタンスを共有する。(これはこれらのアプレット・インスタンスにより所有されるクリア・オン・リセット・タイプ及びクリア・オン・デセレクト・タイプの両方の一時的オブジェクトを含む。)

同じパッケージ内のどんなアプレット・インスタンスにより所有されるクリア・オン・デセレクト・タイプの一時的オブジェクトは、このパッケージ内のいずれかのアプレット・インスタンスが現在選択されたアプレットである時に、アクセス可能である。

【0125】

7. トランザクション及びアトミシテイ

トランザクションは、持続性データの更新の論理的組である。例えば、1つの口座から別の口座にある量のお金を移動することは銀行トランザクションである。トランザクションがアトミックであることが重要である。データ・フィールドの全てが更新されるか、又は全くされないかのいずれかである。J C R Eは、もしトランザクションが正常に終了しなければ、カードのデータはその元のトランザクション前状態に回復されるように、アトミック・トランザクションのための健全な支援を与える。この機構は、トランザクション中の電源喪失やプログラムエラーなどのもしトランザクションの全ステップが正常に終了しない場合にデータを壊すであろう出来事に対して、保護する。

【0126】

7. 1 アトミシテイ

アトミシテイは、1つのオブジェクト又はクラス・フィールド又は配列成分の更新中の停止、故障、又は致命的な例外の後に、持続性記憶の内容をカードがどのように処理するかを定義する。もし電源が更新中に喪失したならば、アプレット開発者は、電源が回復された時にフィールド又は配列成分が含むものに依存することができなければならない。

ジャバ・カード・プラットフォームは、単一の持続性オブジェクト又はクラス・フィールドのどんな更新もアトミックであることを保証する。これに加えて、ジャバ・カード・プラットフォームは、持続性配列に対して、単一成分レベル・アトミシテイを与える。すなわち、もしスマート・カードがC A Dセッションを横断して保存されるべきデータ要素(オブジェクト／クラス内のフィールド又は配列の成分)の更新中に電源を喪失すると、データ要素はその前の値に回復される。

いくつかのメソッドはまた、複数のデータ要素のブロック更新のためのアトミシテイを保証する。例えば、Util.arrayCopyメソッドのアトミシテイは、全てのバイトが正しくコピーされたか、又はそうでなければ宛先配列はその前のバイト値に回復されるかのいずれかを保証する。

アプレットは配列更新に対してアトミシテイを要求しないかもしれない。Util.arrayCopyNonAtomicメソッドは、この目的のために提供される。それは進行中のトランザクションと一緒にコールされた時でさえも、トランザクション・コミット・バッファを使用しない。

【0127】

7. 2 トランザクション

アプレットは、いくつかの異なるオブジェクト内のいくつかの異なるフィールド又は配列

10

20

30

40

50

成分を、アトミカリーに更新する必要があるかもしれない。全ての更新は、正確に且つ整合的に行なわれるか、そうでなければ、全てのフィールド／成分はそれらの前の値に回復されるかのいずれかである。

ジャバ・カード・プラットフォームは、アプレットがJCSYSTEM.beginTransactionメソッドへのコールと共に更新のアトミックな組の始まりを指定することができるトランザクション・モデルを支援する。この点の後の各オブジェクト更新は、条件付きで更新される。フィールド又は配列成分は更新されたかのように見える。フィールド／配列成分を読み返すとその最新の条件付き値が生ずる。しかし、更新はまだコミットされていない。

アプレットが、JCSYSTEM.commitTransactionをコールする時、全て条件付きの更新が持続性記憶にコミットされる。JCSYSTEM.commitTransactionの完了前に電源喪失又はその他のシステム故障が発生すると、全ての条件付更新フィールド又は配列成分はそれらの前の値に回復される。もしアプレットが、内部的問題に出会う又はトランザクションを取消すと決定した場合、それは、JCSYSTEM.abortTransactionをコールすることにより条件付き更新をプログラムの元に戻すことができる。

【0128】

7. 3 トランザクション期間

トランザクションは、JCREがアプレットのセレクト、デセレクト、プロセス、又はインストール・メソッドからの戻りの際にプログラムの制御を再獲得する時に、常に終了する。これは、トランザクションがアプレットのcommitTransactionへのコールでもって、又は、トランザクションの中止（アプレットによりプログラムの元に戻す又はJCREによるデフォルトによるかのいずれか）でもって正常に終了するかにかかわらず、真である。トランザクション中止のさらなる詳細はパラグラフ7. 6を参照。

トランザクション期間は、JCSYSTEM.beginTransactionへのコールとcommitTransactionへのコール若しくはトランザクションの中止の間のトランザクションの一生である。

【0129】

7. 4 ネストされたトランザクション

現在のモデルは、ネストされたトランザクションは可能でないと仮定する。一時には1つのトランザクションだけが存在できる。もしトランザクションが既に進行中に、JCSYSTEM.beginTransactionがコールされると、TransactionExceptionが投げられる。

トランザクションが進行中がどうかを決定することができるように、JCSYSTEM.transactionDepthメソッドが提供される。

【0130】

7. 5 トランザクション故障のリセット又はティア

トランザクションが進行中に、電源が喪失（ティア）し、又は、カードがリセットされ、又は、その他のシステム故障が発生した場合、JCREは前のJCSYSTEM.beginTransactionをコール以降の全てのフィールド及び配列の成分の条件付き更新をそれらの前の値に回復する。

この動作は、JCREにより自動的にカードを電源喪失、リセット、又は故障から回復した後に再初期化する時に、実行される。JCREはそれらのオブジェクトのどれが（もしあれば）条件付きで更新されたかを決定し、それらを回復する。

注—電源喪失又はカード・リセットにより失敗したトランザクション中に生成されたインスタンスにより使用されたオブジェクト空間は、JCREにより回復できる。

【0131】

7. 6 トランザクションの中止

トランザクションはアプレット又はJCREのいずれかにより中止できる。

7. 6. 1 プログラム的中止

もしアプレットが内部的問題に出会い又はトランザクションを取消す決定をした場合、それはJCSYSTEM.abortTransactionをコールすることにより、条件付き更新をプログラムの元に戻すことができる。もしこのメソッドがコールされると、前のJCSYSTEM.beginTransactionのコール以来の全ての条件付き更新されたフィールド又は配列成分はそれらの前

の値に回復され、そして、JCSYSTEM.transactionDepth値は0にリセットされる。

【0132】

7. 6. 2 JCREによる中止

もしアプレットが進行中のトランザクションの際にセレクト、デセレクト、プロセス又はインストール・メソッドから戻ると、JCREは自動的にトランザクションを中止する。もしトランザクションの進行中にセレクト、デセレクト、プロセス、又はインストール・メソッドからの戻りが発生すると、JCREはあたかも例外が投げられたように動作する。

【0133】

7. 6. 3 JCREのクリーンアップ責任

中止されたトランザクションの際に生成されたオブジェクト・インスタンスは、もしこれらのオブジェクトについての全ての参照が探すことができて無効に変換できれば、削除できる。JCREは中止されたトランザクション中に生成されたオブジェクトについての参照は無効の参照に等しいことを保証する。

【0134】

7. 7 一時的オブジェクト

持続性オブジェクトの更新のみがトランザクションに参加する。一時的オブジェクトの更新は、それらが「内部トランザクション」がどうかにかかわらず決して、元に回復されない。

【0135】

7. 8 コミット容量

プラットフォーム資源が制限されるため、トランザクション中に蓄積できる条件付き更新データのバイト数は制限される。ジャバ・カード技術は、実現の際にどれだけのコミット容量が利用可能かを決定するメソッドを与える。コミット能力は、利用可能な条件付きバイト更新の数の上限を表す。マネージメント・オーバーヘッドのために利用できる条件付きバイト更新の実数の数はそれよりも少ない。

トランザクション中に、もしコミット容量を越えると、例外が投げられる。

【0136】

8. APIトピックス

この章のトピックスは、ジャバ・カード2. 1 API 草案2仕様に詳述された要件を補足する。第1トピックはAPDUクラス内に完全に実現されるジャバ・カードI/O機能である。第2トピックはAPI支援ジャバ・カード・セキュリティと暗号である。JCSYSTEM.classはAPIバージョン・レベルをカプセル化する。

API内部のトランザクション

ジャバ・カード2. 1 API仕様で特別に表示されなければ、APIクラスの実現はトランザクションを開始したり、又は進行中であれば、トランザクションの状態を変更したりしない。

API内部の資源の使用

ジャバ・カード2. 1 API仕様で特別に表示されなければ、実現はオブジェクト・インスタンスの所有者が現在選択されたアプレットでない時でさえ、APIインスタンス・メソッドの呼出しを支援する。換言すれば、特に表示されなければ、実現はクリア・オン・デセレクト・タイプの一時的オブジェクトなどの資源を使用しない。

APIクラスにより投げられた例外

API実現により投げられた全ての例外オブジェクトは、一時的JCRE入口点オブジェクトである。一時的JCRE入口点オブジェクトは、クラス変数、インスタンス変数、又は配列成分内に記憶することができない（参照6. 2. 1）。

【0137】

8. 1 APDUクラス

APDUクラスは、ISO7816-4に基づいたカード・シリアル線を横断するI/Oへのアクセスをカプセル化する。APDUクラスは、下にあるI/Oトランスポート・プ

10

20

30

40

50

ロトコルに独立に設計されている。

J C R E は、 $T = 0$ 又は $T = 1$ トランスポート・プロトコル又は両方を支援する。

【0138】

8. 1. 1 外向きデータ転送ため $T = 0$ 詳細

ブロック連鎖機構を支援しない旧式の C A D / ターミナルとの互換性のため、A P D U クラスは、setOutgoingNoChainingメソッドを介してモード選択を可能にする。

【0139】

8. 1. 1. 1 連鎖無しの制約された転送

setOutgoingNoChainingメソッドをコールすることによりアプレットにより要求された出力転送の無連鎖モードの時、以下のプロトコル順序が続く。

注—無連鎖モードが使用される時、waitExtensionメソッドへのコールは理由コード、ILLEGAL_USE、と共に、APDUException、を投げる。

表記

L_e = C A D が期待する長さ。

L_r = setOutgoingLengthメソッドを介して設定されたアプレット返答長さ。

$\langle I N S \rangle$ = 次に転送される全データ・バイトを示す、入力ヘッダー I N S バイトに等しいプロトコル・バイト。

$\langle \sim I N S \rangle$ = 次に転送される約 1 データ・バイトを示す、入力ヘッダー I N S バイトの補数であるプロトコル・バイト。

$\langle S W 1, S W 2 \rangle$ = I S O 7 8 1 6 - 4 内のような応答状態バイト。

I S O 7 8 1 6 - 4 ケース 2

$L_e = L_r$

1. カードが、標準の $T = 0 \langle I N S \rangle$ 又は $\langle \sim I N S \rangle$ 手順バイト機構を使用して、 L_r バイトの出力データを送信する。

2. カードが、Applet.processメソッドの完了時に $\langle S W 1, S W 2 \rangle$ 完了状態を送信する。

$L_r < L_e$

1. カードが、 $\langle 0 x 6 1, L_r \rangle$ 完了状態バイトを送信する。

2. C A D が、 $L_e = L_r$ でもって G E T R E S P O N S E 命令を送信する。

3. カードが、標準の $T = 0 \langle I N S \rangle$ 又は $\langle \sim I N S \rangle$ 手順バイト機構を使用して、 L_r バイトの出力データを送信する。

4. カードが、Applet.processメソッドの完了時に $\langle S W 1, S W 2 \rangle$ 完了状態を送信する。

$L_r > L_e$

1. カードが、標準の $T = 0 \langle I N S \rangle$ 又は $\langle \sim I N S \rangle$ 手順バイト機構を使用して、 L_e バイトの出力データを送信する。

2. カードが、 $\langle 0 x 6 1, (L_r - L_e) \rangle$ 完了状態バイトを送信する。

3. C A D が、新 $L_e \leq L_r$ で持つ G E T R E S P O N S E 命令を送信する。

4. カードが、標準の $T = 0 \langle I N S \rangle$ 又は $\langle \sim I N S \rangle$ 手順バイト機構を使用して、(新) L_e バイトの出力データを送信する。

5. 要求されるならば残りの出力データ・バイト (L_r) を送るのに必要なステップ 2 - 4 を繰返す。

6. カードが、Applet.processメソッドの完了時に $\langle S W 1, S W 2 \rangle$ 完了状態を送信する。

I S O 7 8 1 6 - 4 ケース 4

ケース 4 において、 L_e は以下の初期交換後に決定される。

1. カードが、 $\langle 0 x 6 1, L_r$ 状態バイト $\rangle$ を送信する。

2. C A D が、 $L_e \leq L_r$ を持つ G E T R E S P O N S E 命令を送信する。

プロトコル順序の残りは、上記のケース 2 と同じ。

もし、アプレットが早く中止して L_e バイトよりも少なく送信すると、C A D が期待する

10

20

30

40

50

転送長を満たすためにゼロが代りに送られてもよい。

【0140】

8. 1. 1. 2 通常出力転送

アプレットにより出力転送の無連鎖モードが要求されない時（すなわち、setOutgoing方法が使用される）、以下のプロトコル順序が続く。

どんなISO-7816-3/4準拠T=0プロトコル転送順序が使用できる。

注—sendBytes又はsendBytesLongメソッドの連続したコール間でアプレットにより、waitExtensionメソッドを呼出することができる。waitExtensionメソッドは、0x60手順バイトを使用して追加の作業時間を要求する（ISO-7816-3）。

【0141】

8. 1. 1. 3 追加的T=0要件

何時でも、T=0出力転送プロトコルが使用中で、そしてAPDUクラスがカードからの<0x61, xx>の返答状態に応答してCADからのGETRESPONSE命令を待っている時、もしCADが異なる命令を送れば、sendBytes又はsendBytesLongメソッドは理由コードNO_TO_GETRESPONSEと共にAPDUExceptionを投げる。

この点からのsendBytes又はsendBytesLongメソッドへのコールは、理由コードILLEGAL_USEと共にAPDUExceptionを生ずる。もし、NO_TO_GETRESPONSE例外が投げられた後に、アプレットによりISOExceptionが投げられると、JCRCはその理由コードの返答状態を破棄する。JCRCは新たに受取った命令と共にAPDUを再開し、APDUの発送を再開する。

【0142】

8. 1. 2 外向きデータ転送のためのT=1詳細

8. 1. 2. 1 無連鎖転送の制約

setOutgoingNoChainingメソッドをコールすることによりアプレットにより要求された出力転送の無連鎖モードの時、以下のプロトコル順序が続く。

表記

L e = CADが期待する長さ。

L r = setOutgoingLengthメソッドを介して設定されるアプレット返答長さ。

トランスポート・プロトコル順序ブロック連鎖を使用しない。転送中にMビット（より多くのデータ・ビット）はIブロックのPCB内に設定されない。換言すれば、外向きデータの全体（L r バイト）は1つのIブロック内で転送される。

（もし、アプレットが早く中止してL e バイトよりも少なく送信すると、残りのブロックの長とを満たすためにゼロが代りに送られる。）

注—無連鎖モードが使用される時、waitExtensionメソッドへのコールは理由コード、ILLEGAL_USE、と共に、APDUException、を投げる。

【0143】

8. 1. 2. 2 通常出力転送

アプレットにより出力転送の無連鎖モードが要求されない時、すなわち、setOutgoing方法が使用される時、以下のプロトコル順序が続く。

どんなISO-7816-3/4準拠T=1プロトコル転送順序が使用できる。

注—sendBytes又はsendBytesLongメソッドの連続したコール間でアプレットにより、waitExtensionメソッドを呼出することができる。waitExtensionメソッドは、T=0モード内の追加の1作業時間を要求することに等しい、INFユニットのWTX要求を有するSブロック命令を送る（ISO7816-3参照）。

【0144】

8. 2 セキュリティ及び暗号パッケージ

次のクラス内のgetInstanceメソッドは、要求されたアルゴリズムをコールするアプレットのコンテキスト中に実現インスタンスを戻す。

Javacard.security.MessageDigest

Javacard.security.Signature

Javacard.security.RandomData

Javacardx.crypto.Cipher

J C R Eの実現は、A P Iにリストされたアルゴリズムの0又は複数を実現する。実現されていないアルゴリズムが要求される時、このメソッドは理由コード、NO_SUCH_ALGORITHM、を持った、CryptoException、を投げる。

上記クラスの実現は、対応する基本クラスを拡張し、全てのアブストラクト・メソッドを実現する。実現インスタンスと関連された全てのデータ割当ては

、アプレットのインストレーション中に要求された資源の不足を早期にフラグを立てることができることを保証するためにインスタンス構成の時に実行される。

同様に、javacard.security.keyBuilderクラスのbuildkey方法は、要求されたキー・タイプの実現インスタンスを戻す。J C R Eは、0又は複数のキー・タイプを実現する。実現されていないキー・タイプが要求される時、このメソッドは理由コード、NO_SUCH_ALGORITHM、を持った、CryptoException、を投げる。

キー・タイプの実現は関連したインターフェイスを実現する。キー実現インスタンスに関連した全てのデータ割当ては、アプレットのインストレーション中に要求された資源の不足を早期にフラグを立てることができることを保証するためにインスタンス構成の時に実行される。

【0145】

8. 3 J C S y s t e m C l a s s

ジャバ・カード2. 1において、getVersionメソッドは、0x0201、（短い）を戻す。

【0146】

9. 仮想マシン・トピックス

この章のトピックスは仮想マシンの詳細を説明する。

9. 1 資源故障

回復可能な資源状態（例えば、ヒープ空間）の枯渇は、理由コード、NO_RESOURCE、を持った、SystemException、を生ずる。一時的配列を生成するために使用されるJ C S y s t e m内のフアクトリイ・メソッドは、一時的空間の不足を示すために、理由コード、NO_TRANSIENT_SPACE、を持った、SystemException、を投げる。

スタックのオーバーフローなどの全ての仮想マシン・エラー（回復不可能）は、仮想マシン・エラーを生ずる。これらの状態は仮想マシンを停止させる。このような回復不可能な仮想マシン・エラーが発生する時、実現は選択的にカードをミュートにするか将来の使用を阻止することを要求できる。

【0147】

10. アプレット・インストラー

ジャバ・カード技術を使用したスマート・カード上のアプレット・インストレーションは、複雑なトピックである。ジャバ・カードA P I 2. 1は、それらの実現において可能な限りの多くの自由度を、J C R E実現者に与えることを意図している。しかし、いくつかの基本的な共通仕様が、ジャバ・カード・アプレットがインストールする特定のインストーラの実現の詳細を知ることなくインストールすることを可能にするために、要求される。

この仕様は、インストーラ概念を定義し、そして広い範囲にわたり可能なインストーラの実現の相互操作を達成するために、最小のインストレーション要件を詳細に示す。

アプレット・インストーラはJ C R E 2. 1仕様の選択的な部分である。すなわち、J C R Eの実現は、発行後にインストーラを含むことは必ずしも必要でない。しかし、もし実現されれば、インストーラは節9. 1に詳細に説明された振舞いを支援することが要求される。

【0148】

10. 1 インストーラ

ジャバ・カード技術を使用してスマート・カード上にアプレットをインストールするために必要な機構は、インストーラと呼ばれるカード上の部品に具体化されている。

CADに対して、インストーラはアプレットに見える。それはAIDを持ち、そしてこのAIDがセレクト命令により成功的に処理された時に現在選択されたアプレットとなる。一旦、選択されると、インストーラは他のアプレットと同様に振舞う。

- ・その他の選択されたアプレットと同様に、それは全てのAPDUを受取る。
- ・その設計仕様は、さまざまな前処理下でこれらの命令のセマティックスと共に受取ることで期待されるAPDUのフォーマット及び様々な種類を規定する。
- ・それが受取る全てのAPDUを処理して応答する。不正確なAPDUはある種類のエラー状態を持って応答される。
- ・別のアプレットが選択された時（又はカードがリセットされた時又は電源がカードから取外された時）、インストーラは選択解除されて、それが選択される次の時まで中止される。

10

【0149】

10. 1. 1 インストーラ実現

インストーラは、カード上にアプレットとして実現する必要はない。要件は、インストーラの機能は選択可能であることだけである。この要件の結果として、インストーラ成分は、非インストーラ・アプレットが選択された時又はアプレットが選択されない時、呼出すことができない。

明らかに、JC RE実現者はインストーラをアプレットとして実現できることを選択できる。もしそうならば、インストーラはアプレット・クラスを拡張するためにコードされて、セレクト、プロセス、及びデセレクト・メソッドの呼出しに応答する。

20

しかし、JC RE実現者は、インストーラをそれが外界に対して選択可能な振舞いを与える限り、別の方法で実現できる。この場合、JC RE実現者はAPDUがインストーラ・コード・モジュールに配達される別の機構を与える自由度を有する。

【0150】

10. 1. 2 インストーラAID

インストーラは選択可能であるから、それはAIDを有する。JC RE実現者はそれらのインストーラが選択されるAIDを自由に選択できる。複数のインストーラが実現できる。

【0151】

10. 1. 3 インストーラAPDU

ジャバ・カードAPI 2. 1は、インストーラに対してAPDUを指定しない。JC REは実現者はそれらのインストーラに仕事を指示するためにAPDU命令を完全に自由に選択できる。

30

モデルは、カード上のインストーラがCAD上で走るインストレーション・プログラムにより駆動される。インストレーションが成功するために、このCADインストレーション・プログラムは以下のことが可能でなければならない。

- ・カードを認識する。
- ・カード上のインストーラを選択する。
- ・適当なAPDUをカード・インストーラに送ることにより、インストレーション・プロセスを駆動する。これらのAPDUは、以下を含む。

40

＞インストレーションが認証されることを保証するための、認証情報。

＞カード・メモリ内にロードされるべきアプレット・コード

＞カード上に既に存在するコードとアプレット・コードをリンクするためのリンケージ情報

＞アプレットのインストール・メソッドに送られるべきインスタンス・イニシャライゼーション・パラメータ・データ

ジャバ・カードAPI 2. 1はCADインストレーション・プログラムの詳細を指定せず、また、APDUはそれとインストーラの間に送られない。

【0152】

10. 1. 4 インストーラ振舞い

50

J C R E 実現者は、以下を含むそれらのインストーラの他の振舞いを定義しなければならない。

- ・ インストーレーションが中止できるかどうか、そしてこれがいかになされたか。
- ・ インストーレーション中に、もし例外、リセット、又は電源喪失が発生したら、何が生ずるか。
- ・ インストーラがその仕事を終える前に、もし別のアプレットが選択されたら何が生ずるか。

J C R E は、以下の場合、アプレットが成功的にインストールされたと見なされないことを保証する。

- ・ `Applet.register` メソッドからの成功的な戻りの前に、アプレットのインストール・メソッドが例外を投げる（パラグラフ 9. 2 参照）。 10

【0153】

10. 1. 5 インストーラ特権

インストーラはアプレットとして実現できるが、インストーラは典型的に「他のアプレット」に利用可能ではない特徴にアクセスする必要がある。例えば、J C R E 実現者の実現に依存して、インストーラは次のことが必要である。

- ・ オブジェクト・システム及び／又は標準セキュリティをバイパスして、直接にメモリに読み書きする。
- ・ 他のアプレット又は J C R E により所有されたオブジェクトへのアクセス。
- ・ J C R E の非入口点メソッドを呼出す。
- ・ 新たにインストールされたアプレットのインストール・メソッドを呼出すことができる。

再び、インストトラ実現を決定し、そしてインストーラを支援するのに必要な J C R E 実現内の特徴を供給することは、各 J C R E 実現者の責任である。J C R E 実現者は、普通のアプレットに利用できないようにするため、そのような特徴のセキュリティに責任を有する。

【0154】

10. 2 新にインストールされたアプレット

インストーラとインストールされているアプレットの間に単一のインターフェイスが存在する。インストーラが正しく実行（ローディング及びリンキングなどの実行されたステップ）するためのアプレットを準備した後に、インストーラはアプレットのインストール・メソッドを呼出す。このメソッドはアプレット・クラスと定義される。 30

インストーラからアプレットのインストール・メソッドが呼出される正確な機構は、J C R E 実現者定義の実現詳細である。しかし、インストール・メソッド（新しいオブジェクトを生成するなど）により実行されたコンテキストに関係した操作が新アプレットのコンテキスト内でなされ、そしてインストーラのコンテキスト内でないようにするため、コンテキスト切換えがある。インストーラはアプレット・クラス初期化（`<clinit>`）メソッド中に生成された配列オブジェクトが、新アプレットのコンテキストにより所有されることも保証する。

もし全てのステップが失敗又は例外が投げられることなく、実行する `Applet.register` メソッドからの成功的な戻りを含むまで、完了した場合、アプレットのインストールは完全とみなされる。その点において、インストールされたアプレットは選択可能である。 40

パラメータ・データの最大サイズは 32 バイトである。セキュリティの理由のため、`bArray` パラメータは戻りの後にゼロにされる（APDU バッファがアプレットのプロセス・メソッドからの戻りの際にゼロにされるのと同様）。

【0155】

10. 2. 1 インストーレーション・パラメータ

32 バイトの最大サイズ以外に、ジャバ・カード API 2. 1 インストーレーション・パラメータ・バイト配列セグメントの内容について何ら指示しない。こりはアプレット設計 50

者により完全に定義され、所望のどんなフォーマットでできる。これに加え、これらインスタレーション・パラメータはインストーラに不透明であることを意図している。

CAD内で実行されるインスタレーション・プログラムがインストーラに配達されるべき任意のバイト配列を指定することを可能にするように、JC RE 実現者はそれらのインストーラを設計すべきである。インストーラは、単にこのバイト配列を `b A r r a y` パラメータ内の目標のアプレットのインストール・メソッドへ転送する。典型的な実現は、セマンティック「付随するバイト配列の内容を送るアプレットのインストール・メソッドをコール」を有する JC RE 実現者専有 `A P D U` 命令を定義する。

【0156】

11. API 定数

いくつかの API クラスは、ジャバ・カード API 2.1 レファレンス内にそれらの定数を指定された値を有しない。もし、一定の値がこの JC RE 2.1 仕様の実現者により一貫して指定されない場合、業界全体の互換性は不可能である。この章はジャバ・カード API 2.1 レファレンス内で指定されない必要な一定の値を与える。

【0157】

【表1】

Class `javacard.framework.APDU`

```
public static final byte PROTOCOL_T0 = 0;
public static final byte PROTOCOL_T1 = 1;
```

Class `javacard.framework.APDUException`

```
public static final short ILLEGAL_USE = 1;
public static final short BUFFER_BOUNDS = 2;
public static final short BAD_LENGTH = 3;
public static final short IO_ERROR = 4;
public static final short NO_T0_GETRESPONSE = 0xAA;
```

Interface `javacard.framework.ISO7816`

```
public final static short SW_NO_ERROR = (short)0x9000;
public final static short SW_BYTES_REMAINING_00 = 0x6100;
public final static short SW_WRONG_LENGTH = 0x6700;
public final static short SW_SECURITY_STATUS_NOT_SATISFIED = 0x6982;
public final static short SW_FILE_INVALID = 0x6983;
public final static short SW_DATA_INVALID = 0x6984;
public final static short SW_CONDITIONS_NOT_SATISFIED = 0x6985;
public final static short SW_COMMAND_NOT_ALLOWED = 0x6986;
public final static short SW_APPLET_SELECT_FAILED = 0x6999;
public final static short SW_WRONG_DATA = 0x6A80;
public final static short SW_FUNC_NOT_SUPPORTED = 0x6A81;
public final static short SW_FILE_NOT_FOUND = 0x6A82;
public final static short SW_RECORD_NOT_FOUND = 0x6A83;
public final static short SW_INCORRECT_P1P2 = 0x6A86;
public final static short SW_WRONG_P1P2 = 0x6B00;
public final static short SW_CORRECT_LENGTH_00 = 0x6C00;
public final static short SW_INS_NOT_SUPPORTED = 0x6D00;
public final static short SW_CLA_NOT_SUPPORTED = 0x6E00;
public final static short SW_UNKNOWN = 0x6F00;
public static final short SW_FILE_FULL = 0x6A84;
public final static byte OFFSET_CLA = 0;
public final static byte OFFSET_INS = 1;
public final static byte OFFSET_P1 = 2;
public final static byte OFFSET_P2 = 3;
```

【0158】

【表2】

10

20

30

40

```

public final static byte OFFSET_LC = 4;
public final static byte OFFSET_CDATA = 5;
public final static byte CLA_ISO7816 = 0x00;
public final static byte INS_SELECT = (byte) 0xA4;
public final static byte INS_EXTERNAL_AUTHENTICATE = (byte) 0x82;

```

Class javacard.framework.JCSystem

```

public static final byte NOT_A_TRANSIENT_OBJECT = 0;
public static final byte CLEAR_ON_RESET = 1;
public static final byte CLEAR_ON_DESELECT = 2;

```

Class javacard.framework.PINException

```

public static final short ILLEGAL_VALUE = 1;

```

10

Class javacard.framework.SystemException

```

public static final short ILLEGAL_VALUE = 1;
public static final short NO_TRANSIENT_SPACE = 2;
public static final short ILLEGAL_TRANSIENT = 3;
public static final short ILLEGAL_AID = 4;
public static final short NO_RESOURCE = 5;

```

Class javacard.security.CryptoException

```

public static final short ILLEGAL_VALUE = 1;
public static final short UNINITIALIZED_KEY = 2;
public static final short NO_SUCH_ALGORITHM = 3;
public static final short INVALID_INIT = 4;
public static final short ILLEGAL_USE = 5;

```

Class javacard.security.KeyBuilder

20

```

public static final byte TYPE_DES_TRANSIENT_RESET = 1;
public static final byte TYPE_DES_TRANSIENT_DESELECT = 2;
public static final byte TYPE_DES = 3;
public static final byte TYPE_RSA_PUBLIC = 4;
public static final byte TYPE_RSA_PRIVATE = 5;
public static final byte TYPE_RSA_CRT_PRIVATE = 6;
public static final byte TYPE_DSA_PUBLIC = 7;
public static final byte TYPE_DSA_PRIVATE = 8;
public static final short LENGTH_DES = 64;
public static final short LENGTH_DES3_2KEY = 128;
public static final short LENGTH_DES3_3KEY = 192;
public static final short LENGTH_RSA_512 = 512;
public static final short LENGTH_RSA_768 = 768;
public static final short LENGTH_RSA_1024 = 1024;
public static final short LENGTH_RSA_2048 = 2048;
public static final short LENGTH_DSA_512 = 512;
public static final short LENGTH_DSA_768 = 768;
public static final short LENGTH_DSA_1024 = 1024;

```

30

Class javacard.security.MessageDigest

```

public static final byte ALG_SHA = 1;
public static final byte ALG_MD5 = 2;
public static final byte ALG_RIPEMD160 = 3;

```

Class javacard.security.RandomData

```

public static final byte ALG_PSEUDO_RANDOM = 1;
public static final byte ALG_SECURE_RANDOM = 2;

```

Class javacard.security.Signature

```

public static final byte ALG_DES_MAC4_NOPAD = 1;
public static final byte ALG_DES_MAC8_NOPAD = 2;
public static final byte ALG_DES_MAC4_ISO9797_M1 = 3;

```

40

【 0 1 5 9 】

【表 3】

```

public static final byte ALG_DES_MAC8_ISO9797_M1 = 4;
public static final byte ALG_DES_MAC4_ISO9797_M2 = 5;
public static final byte ALG_DES_MAC8_ISO9797_M2 = 6;
public static final byte ALG_DES_MAC4_PKCS5 = 7;
public static final byte ALG_DES_MAC8_PKCS5 = 8;
public static final byte ALG_RSA_SHA_ISO9796 = 9;
public static final byte ALG_RSA_SHA_PKCS1 = 10;
public static final byte ALG_RSA_MD5_PKCS1 = 11;
public static final byte ALG_RSA_RIPEMD160_ISO9796 = 12;
public static final byte ALG_RSA_RIPEMD160_PKCS1 = 13;
public static final byte ALG_DSA_SHA = 14;
public static final byte MODE_SIGN = 1;
public static final byte MODE_VERIFY = 2;

```

10

Class javacardx.crypto.Cipher

```

public static final byte ALG_DES_CBC_NOPAD = 1;
public static final byte ALG_DES_CBC_ISO9797_M1 = 2;
public static final byte ALG_DES_CBC_ISO9797_M2 = 3;
public static final byte ALG_DES_CBC_PKCS5 = 4;
public static final byte ALG_DES_ECB_NOPAD = 5;
public static final byte ALG_DES_ECB_ISO9797_M1 = 6;
public static final byte ALG_DES_ECB_ISO9797_M2 = 7;
public static final byte ALG_DES_ECB_PKCS5 = 8;
public static final byte ALG_RSA_ISO14088 = 9;
public static final byte ALG_RSA_PKCS1 = 10;
public static final byte MODE_DECRYPT = 1;
public static final byte MODE_ENCRYPT = 2;

```

20

【0160】

用語集

A I Dは、I S O 7 8 1 6 - 5に定義されたアプリケーション識別子の略。

A P D Uは、I S O 7 8 1 6 - 4に定義されたアプリケーション・プロトコル・データ・ユニットの略。

A P Iは、アプリケーション・プログラム・インターフェイスの略。a p iはアプリケーション・プログラムがオペレーティング・システム又はその他のサービスをアクセスするときに呼ぶ約束事を定義する。

アプレットは、この書類の文脈内では、ジャバ・カード技術内の選択、コンテキスト、機能、及びセキュリティの基本的単位である、ジャバ・カード・アプレットである。

30

アプレット開発者は、ジャバ・カード技術仕様を用いてジャバ・カード・アプレットを生成する人を言う。

アプレット・フアイヤウォールは、V Mが他のアプレット・コンテキスト又はJ C R Eコンテキストにより所有されたオブジェクトへのアプレットの認証されないアクセスから防止し、そして違反を報告し又は対処するジャバ・カード技術内の機構。

アトミック操作は、その全体が完了するか（もし操作が成功すると）、又は操作のどの部分も全く完了しない（もし操作が失敗したら）のいずれかである操作である。

アトミシテイは、特定の操作がアトミックであるか否か、そして電源が喪失、又はカードがC A Dから予期されずに取外された場合の適当なデータ回復が必要な操作を言う。

A T Rは、リセットの返答の略。A T Rは、リセット条件後にジャバ・カードにより送られる一連のバイトである。

40

C A Dは、カード受入装置の略。C A Dはカードが挿入される装置である。

キャストは、1つのデータ・タイプから別のタイプへの明示的な変換である。

c J C Kは、ジャバ・カード技術仕様の実現の許容性を確認するためのテストの一揃い。

c J C Kはテストの一揃いを実行するためジャバ・テスト・ツールを使用する。

クラスは、オブジェクト指向言語のオブジェクトの原型（プロトタイプ）である。クラスは共通の構造と振舞いを共有する1組のオブジェクトとみなすことができる。クラスの構造は、そのクラスのオブジェクトの状態を表すクラス変数により定義され、その振舞いはクラスに関連する1組のメソッドにより与えられる。

クラスは、クラス階層に関係する。1つのクラスは他のクラス（そのスーパークラス）の

50

特別化（サブクラス）であってよい。それは他のクラスへの参照を有し、そしてそれは他のクラスをクライアント－サーバー関係で使用してもよい。コンテキスト（アプレット実行コンテキスト参照）。

現在活動的なコンテキスト。J C R Eは現在活動的なジャバ・カード・アプレット・コンテキストの記録を残す。仮想メソッドがオブジェクト上で呼出され、そしてコンテキスト切換えが必要とされ、そして許可される時、現在活動的なコンテキストはそのオブジェクトを所有するアプレット・コンテキストに対応するために変化される。そのメソッドが戻る時、前のコンテキストが回復される。静的メソッドの呼出しは現在活動的なコンテキストについて影響を与えない。オブジェクトの現在活動的なコンテキスト及び共有状態と一緒に、オブジェクトへのアクセスが許可されるかどうかを決定する。

10

現在選択されたアプレット。J C R Eは現在選択されたジャバ・カード・アプレットの記録を残す。このアプレットのA I Dを有する選択命令を受取る際、J C R Eはこのアプレットを現在選択されたアプレットとする。J C R Eは全てのA P D U命令を現在選択されたアプレットに送る。

E E P R O Mは、電氣的消去可能プログラム可能読み出し専用メモリの略。

ファイヤウォール（アプレット・ファイヤウォール参照）。

フレームワークは、A P Iを実現する1組のクラス。これはコア及び拡張パッケージを含む。責任は、A P D Uの発送、アプレット選択、アトミシテイ管理、及びアプレット・インストーリングを含む。

ガーベッジ・コレクションは、プログラムの実行時に自動的に再クレームされる動的に割当てられた記憶によるプロセスを言う。

20

インスタンス変数はまたフィールドとして知られていて、オブジェクトの内部状態の一部を表す。各オブジェクトはそれ自身のインスタンス変数の組を有する。同じクラスのオブジェクトは同じインスタンス変数を有するが、各オブジェクトは異なった値を持つことができる。

インスタンシエイションとは、オブジェクト指向プログラミングにおいて、そのクラス・テンプレートから特定のオブジェクトを生成することを意味する。これはテンプレートにより指定されたタイプを持つデータ構造の割当てと、デフォルト値又はクラス建設関数により与えられた値でインスタンス変数を初期化することを含む。

J A Rは、ジャバ・アーキテクチャの略。J A Rは多くのファイルを1つに結合するプラットフォームに依存しないファイル・フォーマットである。

30

ジャバ・カード実行時環境（J C R E）は、ジャバ・カード仮想マシンねフレームワーク、及び関連のネイティブ・メソッドからなる。

J C 2 1 R Iは、ジャバ・カード2. 1レファレンス実行の略。

J C R E実現者は、ジャバ・カードA P Iを使用してベンダー固有の実現を作成する人。

J C V Mは、ジャバ・カード仮想マシンの略。J C V Mは、O Pカード・アーキテクチャの基礎である。J C V Mは、バイトコードを実行し、そしてクラス及びオブジェクトを管理する。これはアプリケーション間の分離（ファイヤウォール）を実施し及び安全なデータ共有を可能にする。

J D Kは、ジャバ開発キットの略。J D Kはジャバでプログラムするために必要な環境を与えるサン・マイクロシステムズ社の製品。J D Kはさまざまなプラットフォームで利用可能であるが、最も有名なのは、サン・ソラリス及びマイクロソフト・ウィンドウズ（商標）である。

40

メソッドは、オブジェクト指向言語において、1つ又は複数のクラスと関連した手順若しくはルーチンに与えられた名前である。

名前空間は、全ての名前が独特な1組の名前。

オブジェクト指向は、メソッドと呼ばれるデータを操作する1組のルーチンによりカプセル化されたデータ構造である、オブジェクトに基づいたプログラミング方式である。

オブジェクトは、オブジェクト指向プログラミングにおいて、そのクラスにより与えられたテンプレートに従い定義されたデータ構造の独特なインスタンスである。各オブジェク

50

トは、そのクラスに属する変数についてそれ自身の値を有し、そのクラスにより定義されたメッセージ（メソッド）に応答できる。

パッケージは、ジャバ・プログラミング言語内の名前空間であり、クラス及びインターフェイスを有することができる。パッケージはジャバ・プログラミング言語内の最小ユニットである。

持続性オブジェクトは、持続性オブジェクトとそれらの値が、無限に、1つのC A Dセッションから次に持続するものであるオブジェクトはデフォルトで持続性である。持続性オブジェクトの値はトランザクションによりアトミカリイに更新される。持続性の語はカード上にオブジェクト指向データベースが存在することを意味しない。又、オブジェクトが直列化／非直列化されることを意味しない。カードが電源を喪失する時にオブジェクトが失われないことを意味する。

10

共有可能インターフェイスは、共有されたインターフェイス・メソッドの1組を定義する。これらのインターフェイス・メソッドは、それらを実現するオブジェクトが別のアプレット・コンテキストにより所有されている時、1つのアプレット・コンテキストにより呼出することができる。

共有可能インターフェイス・オブジェクト（S I O）は、共有可能インターフェイスを実現するオブジェクトである。

トランザクションは、開発者がプログラム・コード内にトランザクションの開始及び終了を指示することにより操作の広がり定義する、アトミック操作である。

一時的オブジェクト。一時的オブジェクトの値は、1つのC A Dセッションから次のセッションに持続せず、特定の間隔でデフォルト値にリセットされる。一時的オブジェクトの値の更新はアトミックではなく、トランザクションにより影響されない。

20

【0161】

1/5/99 12:49PM Havnor: Stuff: JCRE D2 14DEC98: READ-ME-JCRE21-DF2.txt Page 1

日付： 1998年12月16日

ジャバ・カード・ライセンシー殿

J C R E 2 1 - D F 2 - 1 4 D E C 9 8 . z I P は、ライセンシーの検討と意見のために、1998年12月14日の日付のジャバ・カード2. 1実行時環境仕様の第2草案を含む。我々は今まで受取った検討フィードバックに基づいて書類を明確に組み込んで来た。Z I P のアーカイブの完全の内容は次の通りである。

30

R E A D - M E - J C R E 2 1 - D E 2 . t x t この R E A D M E 文書ファイル。

J C R E 2 1 - D F 2 . p d f 「ジャバ・カード実行時環境（J C R E）2. 1仕様」の P D F フォーマット。

J C R E 2 1 - D F 2 - c h a n g e b a r . p d f 一見やすくするために前のバージョンから変化バーを有する改訂バージョン。

変更の要約

1. これは第2草案のリリースであり、公開のウェブ・サイトに直ぐに公開される。
 2. 認証されないアクセスを制限するために、一時的 J C R E 入口点オブジェクトの新しい記述が導入された。ファイヤウォール章6. 2. 1
 3. グローバル配列が一時的 J C R E 入口点オブジェクトと似たセキュリティ関係の制限を追加する。ファイヤウォール章6. 2. 2
 4. 一時的 J C R E 入口点オブジェクト及びグローバル配列の記憶に関して、バイトコードの詳細な説明が追加された。章6. 2. 8
 5. 章8に J C R E 所有の例外オブジェクトの一般的説明が追加された。
 6. 一時的ファクトリイ・メソッド内の仮想マシン資源故障の修正された記述。章9. 1
- 「ジャバ・カード2. 1実行時環境仕様」は、ジャバ・カード A P T 2. 1 及びジャバ・カード2. 1仮想マシン仕様書類で言及されるような、ジャバ・カード2. 1実現を完成するための最小の振舞いと実行時環境を詳述する。この仕様はジャバ・カード・アプレットの動作と互換性を保証する必要がある。この仕様書類の目的は、ジャバ・カード2. 1仕様の一揃いの部分として、簡潔な方法で全ての J C R E 要素を一緒にすることである。

40

50

【0162】

検討したコメントを<javaoem-javacard@sun.com>又は下の私のアドレスに送って下さい。
ジャバ・カード・チームとして、ご意見を待っています。

よろしく。

ゴットフライ・デジオルジ

Godfrey DiGiorgi - godfrey.digiorgi@eng.sun.com

OEM Licensee Engineering

Sun Microsystems / Java Software

+1 408 343-1506 FAX +1 408 517-5460

【図面の簡単な説明】

10

【図1】スマート・カードのカード受入装置を装備されたカード受入装置と一緒に使用するためのコンピュータを示す図。

【図2】ネットワークに接続されカード受入装置を装備されたコンピュータを示す図。

【図3】従来技術のスマート・カードなどの小面積装置の例示的なハードウェア・アーキテクチャを示す図。

【図4】従来技術で使用する原理によりアクセスされるオブジェクトを示す図。

【図5】本発明のさまざまな実施の形態を説明するのに使用できる例示的なセキュリティ・モデルを示す図。

【図6】本発明の1つの観点によるコンテキスト障壁又はファイヤ・ウォールによる実行コンテキストの分離を示すブロック図。

20

【図7】本発明を実行するのに有用なソフトウェア・アーキテクチャを示す図。

【図8】本発明の1つの観点によるファイヤ・ウォールを実現するセキュリティ増強プロセスのフロー・チャート。

【図9】本発明の1つの観点によるファイヤ・ウォールを横断するオブジェクトのアクセスを示すブロック図。

【図10】ファイヤ・ウォールを横断するカスケードされたオブジェクトを示すブロック図。

【図11】1つのコンテキスト中の主成分によるファイヤ・ウォールを横断して別のコンテキストへのアクセスを許可するプロセスのフロー・チャート。

【図12】ファイヤ・ウォールを横断したアクセスを許可するための入口点オブジェクトの使用を声明するブロック図。

30

【図13】ファイヤ・ウォールを横断するアクセスのための配列などのグローバル・データ構造の使用を説明するブロック図。

【図14】ファイヤ・ウォールを横断するアクセスを許可するためのスーパー・コンテキストの使用を説明するブロック図。

【図15】ファイヤ・ウォールを横断するアクセスを許可するための共有可能インターフェイス・オブジェクトの使用を説明するブロック図。

【図16】ファイヤ・ウォールを横断したアクセスを許可するセキュリティ増強プロセスのフロー・チャート。

【図17】ブロック1620の詳細を示す図16のフローチャート。

40

【図18】図17のブロック1629の例示的実現を示すフローチャート。

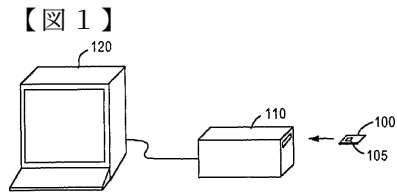


FIG. 1

【図 2】

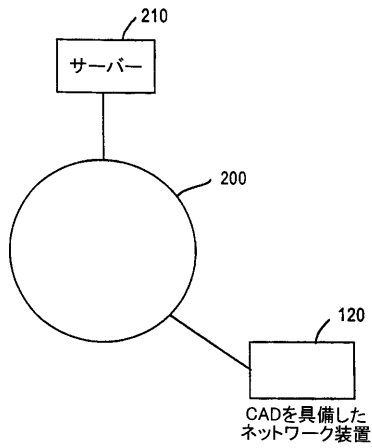
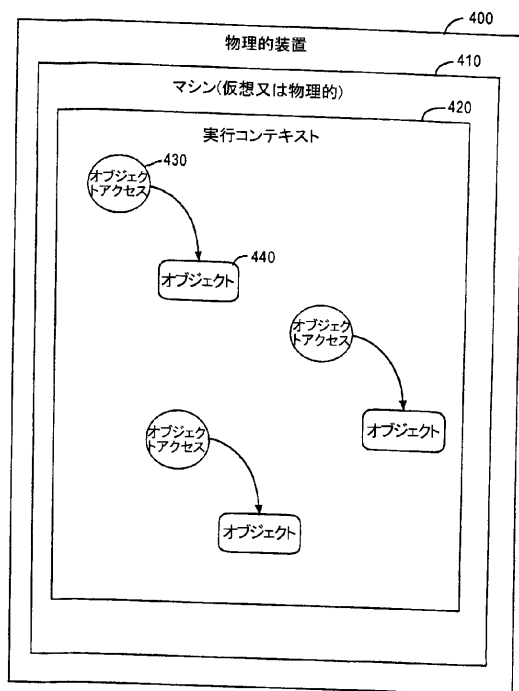
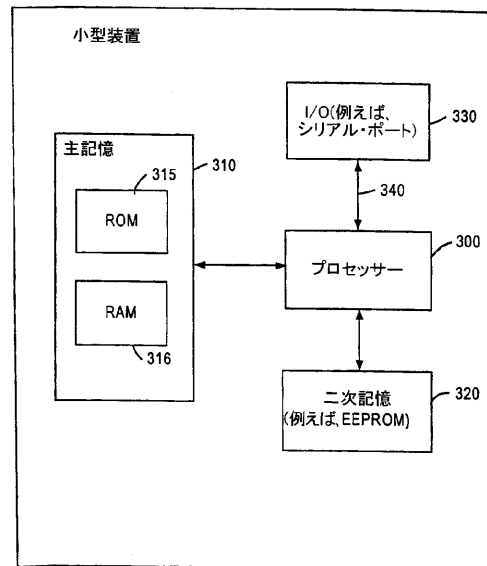


FIG. 2

【図 4】

FIG. 4
(従来技術)

【図 3】

FIG. 3
(従来技術)

【図 5】

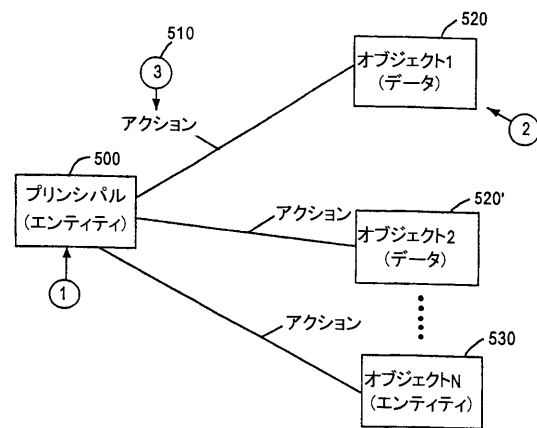


FIG. 5

【図 6】

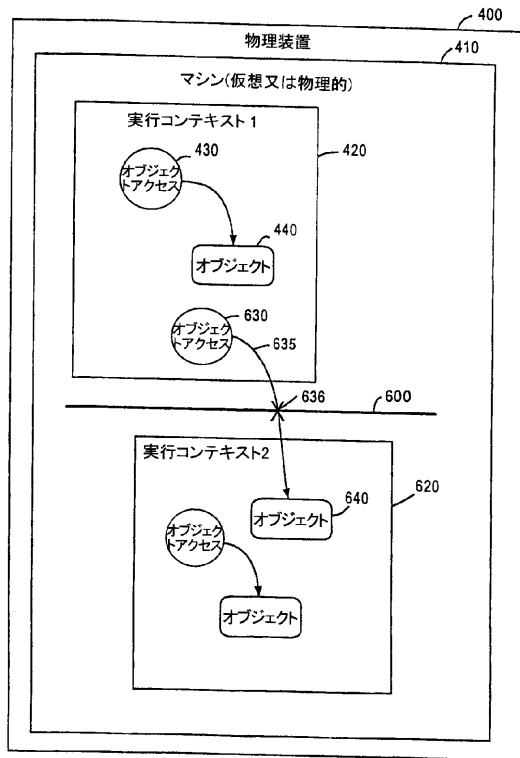


FIG. 6

【図 7】

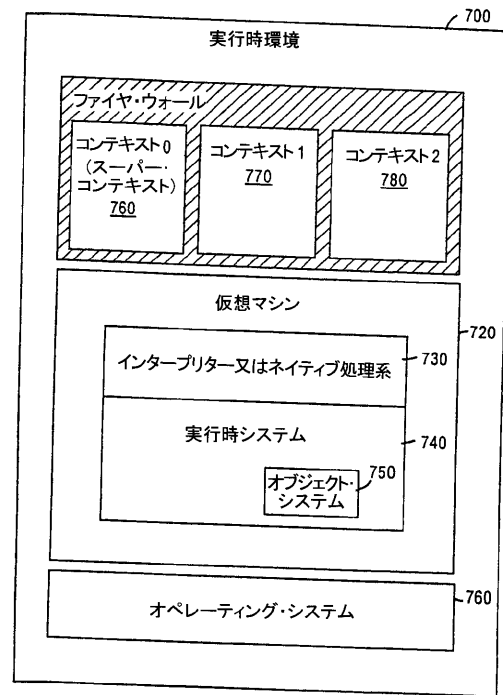


FIG. 7

【図 8】

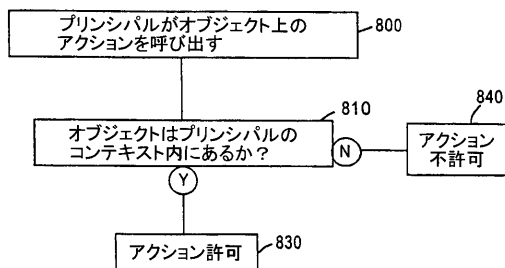


FIG. 8

【図 9】

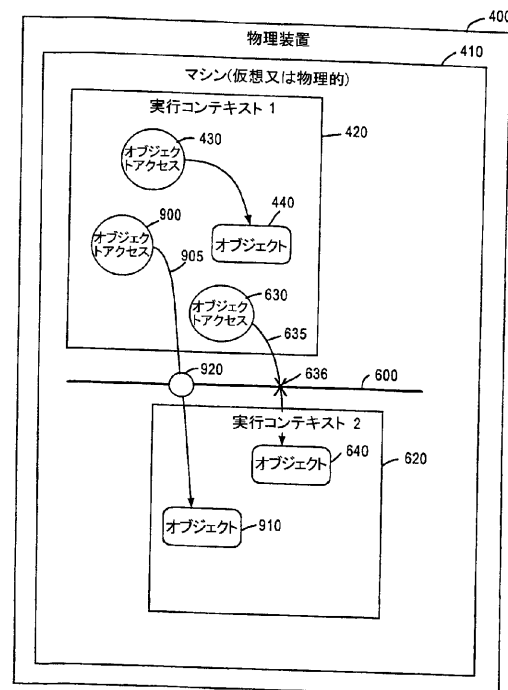


FIG. 9

【図 10】

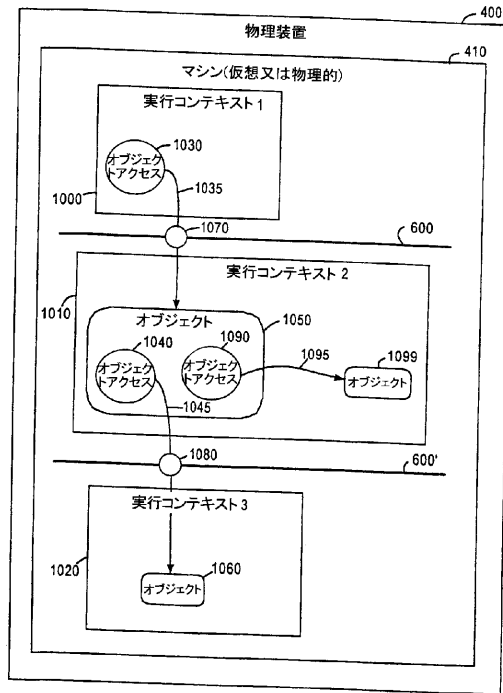


FIG. 10

【図 11】

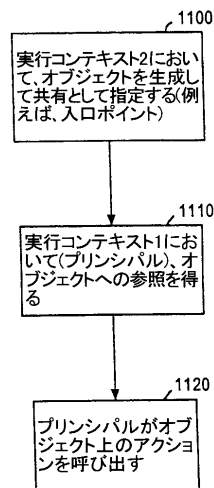


FIG. 11

【図 12】

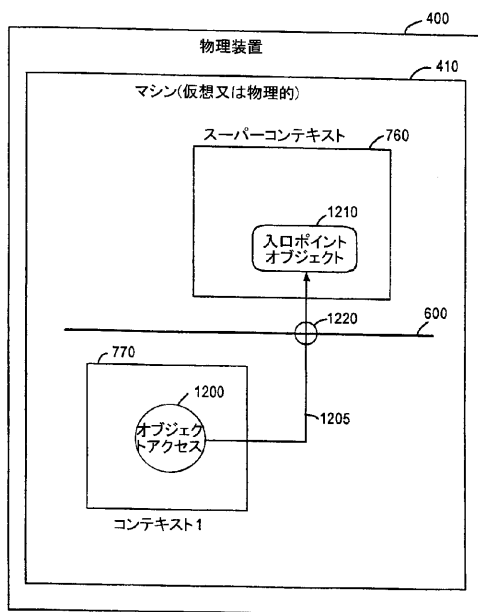


FIG. 12

【図 13】

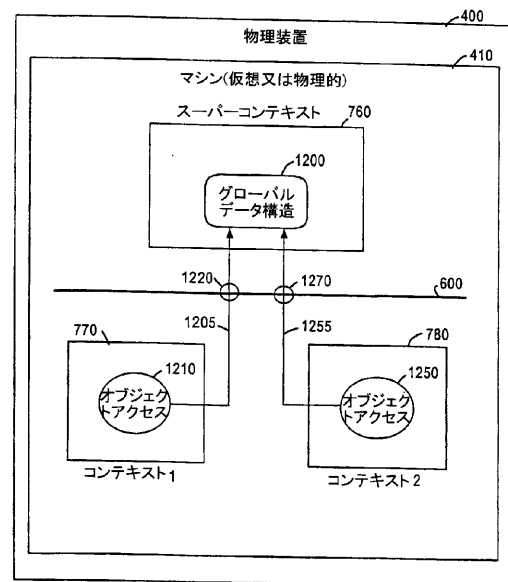


FIG. 13

【図 14】

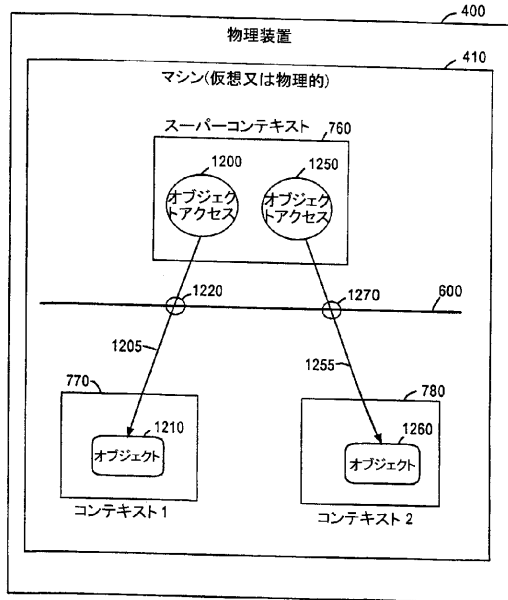


FIG. 14

【図 15】

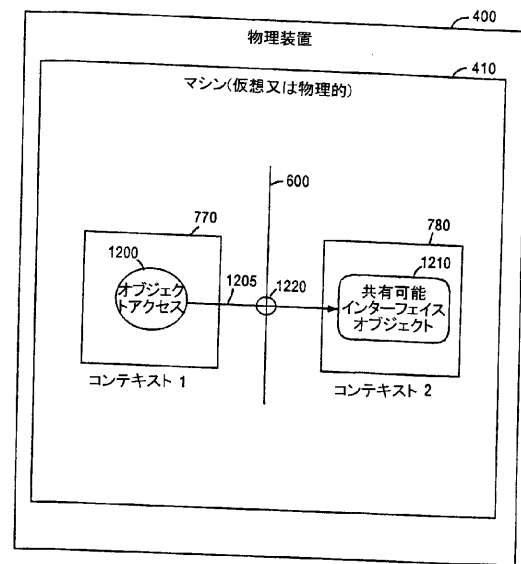


FIG. 15

【図 16】

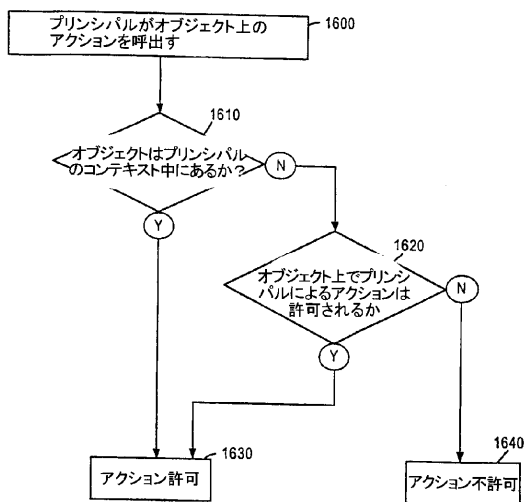


FIG. 16

【図 17】

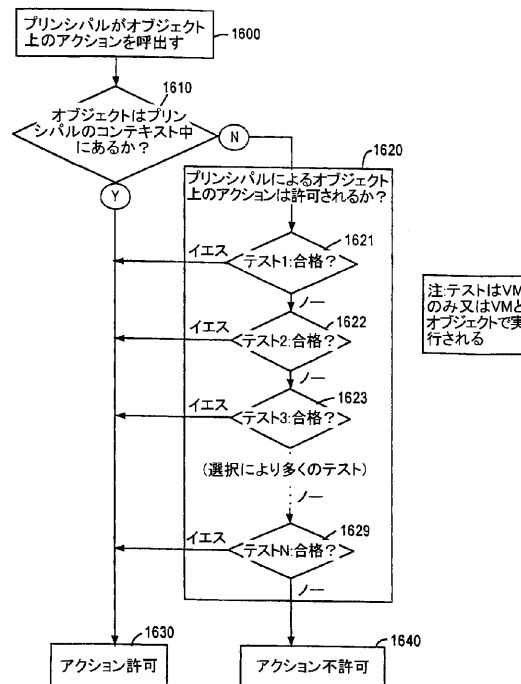


FIG. 17

【図 18】

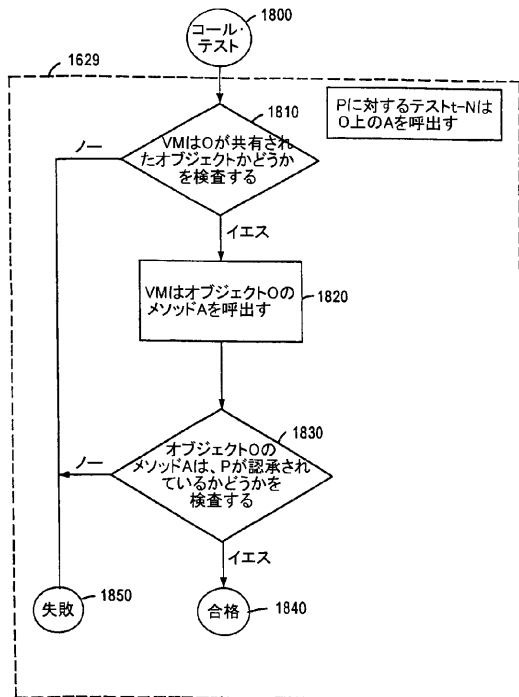


FIG. 18

フロントページの続き

- (72)発明者 バトラー ミッチェル ビー
アメリカ合衆国 カリフォルニア州 94087 サニーヴェイル ノース カスケイド テラス
522
- (72)発明者 ストレイチ アンディー
アメリカ合衆国 カリフォルニア州 94404 フォスター シティー ビーチ パーク ブー
ルヴァード 693

審査官 井上 宏一

- (56)参考文献 特開平05-224956 (JP, A)
特開平09-223200 (JP, A)
特開平10-116293 (JP, A)
特開平07-319714 (JP, A)
特表平09-508733 (JP, A)
国際公開第98/19237 (WO, A1)
米国特許第5204897 (US, A)

(58)調査した分野(Int.Cl., DB名)

G06F 9/46 -9/54
G06F 12/14
G06F 21/22
G06K 19/07