(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-242010
(P2007-242010A)
(43)公開日　平成19年9月20日(2007.9.20)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 11/34	(2006.01)	G06F 11/34	B	5B042

審査請求 未請求　請求項の数 12　ＯＬ　（全 10 頁）

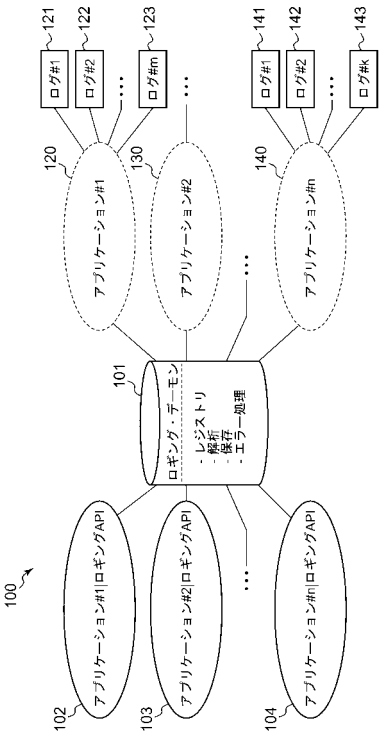
(21)出願番号　特願2007-40442(P2007-40442)
(22)出願日　平成19年2月21日(2007.2.21)
(31)優先権主張番号　11/367666
(32)優先日　平成18年3月3日(2006.3.3)
(33)優先権主張国　米国(US)

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】共通ロギングにより収集される情報をロギングする解析コンピュータ・システム

(57)【要約】
【課題】　コンピュータ・システムに関する複数のロギング・アプリケーションからの情報をロギングするための方法が提供される。
【解決手段】　ロギング・アプリケーションを識別するようにレジストリが構成されており、複数のロギング・アプリケーションは、共通プロトコルおよびレジストリに従ってロギング済み情報を識別するように構成されている。ロギング制御部が、ロギング済み情報を解析するように構成されているとともに、ロギング済み情報を解析に従って保存するように構成されている。レジストリは、ロギング・アプリケーションに従って更新されるように構成されている。ロギング制御部が、ロギング済み情報を複数のロギング・アプリケーションに従って解析するように構成されている。
【選択図】　図１



【特許請求の範囲】
【請求項１】
ロギング・アプリケーションを識別するように構成されているレジストリと、
各々がコンピュータ・システムに関する情報をロギングするように構成され、かつ、前記ロギング済み情報を共通プロトコルおよび前記レジストリに従って識別するように構成されている複数のロギング・アプリケーションと、
前記ロギング済み情報を解析するように構成され、かつ、前記ロギング済み情報を前記解析に従って保存するように構成されているロギング制御部と、
を含むコンピュータ・システム。
【請求項２】
前記レジストリが、ロギング・アプリケーションに従って更新されるように構成されている、請求項１記載のコンピュータ・システム。
【請求項３】
前記ロギング制御部が、前記ロギング済み情報を複数のロギング・アプリケーションに従って解析するように構成されている、請求項１記載のコンピュータ・システム。
【請求項４】
前記ロギング制御部が、前記保存されたロギング済み情報の少なくとも一部を前記複数のロギング・アプリケーションの少なくとも一部の中で共有するように構成されている、請求項３記載のコンピュータ・システム。
【請求項５】
ロギング・エラーを検出した結果として、前記ロギング済み情報を保存するための新規のスペースを割り当てるように構成されているエラー・リソース割当て機能部をさらに含む、請求項１記載のコンピュータ・システム。
【請求項６】
各々がコンピュータ・システムに関する情報をロギングするように構成されている複数のロギング・アプリケーションからの情報をロギングするように構成されているロギング・デーモンであって、
ロギング・アプリケーションを識別するように構成されているレジストリを含み、それによって前記複数のロギング・アプリケーションが前記ロギング済み情報を共通プロトコルおよび前記レジストリに従って識別することができ、さらに、
前記ロギング済み情報を解析するように構成され、かつ、前記ロギング済み情報を前記解析に従って保存するように構成されているロギング制御部を含む、
ロギング・デーモン。
【請求項７】
各々がコンピュータ・システムに関する情報をロギングするように構成されている複数のロギング・アプリケーションからの情報をロギングするための方法であって、
ロギング・アプリケーションを識別するように構成されているレジストリを提供するステップを含み、それによって前記複数のロギング・アプリケーションが前記ロギング済み情報を共通プロトコルおよび前記レジストリに従って識別することができ、さらに、
前記ロギング済み情報を解析するステップと、
前記ロギング済み情報を前記解析に従って保存するステップと、
を含む方法。
【請求項８】
前記レジストリを提供する前記ステップが、前記レジストリをロギング・アプリケーションに従って更新するステップをさらに含む、請求項７記載の方法。
【請求項９】
前記ロギング済み情報を解析する前記ステップが、複数のロギング・アプリケーションに従って実行される、請求項７記載の方法。
【請求項１０】
前記ロギング済み情報を保存する前記ステップが、前記保存されたロギング済み情報の少

なくとも一部を前記複数のロギング・アプリケーションの少なくとも一部の中で共有するステップをさらに含む、請求項９記載の方法。
【請求項１１】
ロギング・エラーを検出した結果として、前記ロギング済み情報を保存するための新規のスペースを割り当てるステップをさらに含む、請求項７記載の方法。
【請求項１２】
各々がコンピュータ・システムに関する情報をロギングするように構成されている複数のロギング・アプリケーションからの情報をロギングするためにコンピュータに、
ロギング・アプリケーションを識別するように構成されているレジストリを提供し、それによって前記複数のロギング・アプリケーションが前記ロギング済み情報を共通プロトコルおよび前記レジストリに従って識別するステップと、
前記少なくとも１つのプログラマブル・コンピュータに、前記ロギング済み情報を解析するステップと、
前記少なくとも１つのプログラマブル・コンピュータに、前記ロギング済み情報を前記解析に従って保存させるステップ、
を実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はコンピュータ・システムに関連する情報のロギングに関し、より詳細には、コンピュータ・システム用の複数のロギング・アプリケーションに関する。
【背景技術】
【０００２】
コンピュータ・システムにおいてアプリケーションの実行ログを保存することは、システムの動作履歴を理解してシステムの性能を分析し、システム障害の徴候を診断する等にとって信頼できる方法である。システムに問題が生じた場合、ロギング済みデータを分析して、問題は何か、原因は何であったかを判断し、問題を解決することができる。ロギング済みデータは、特定の不具合によって引き起こされるシステムの機能停止を最小にするためのツールとすることができる。またロギング済みデータは、アプリケーション開発者が製品をデバッグしてより良い製品を提供するためのツールとすることもできる。ロギング方法は、さもなくばシステムにより使用されるはずのシステム・リソースを犠牲にしている。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
コンピュータ・システムに関する複数のロギング・アプリケーションからの情報をロギングするためのコンピュータ・システム、ロギング・デーモン（logging daemons）、方法、およびコンピュータ・プログラムが提供される。
【課題を解決するための手段】
【０００４】
本発明の１実施形態において、レジストリがロギング・アプリケーションを識別するように構成されており、複数のロギング・アプリケーションは、ロギング済み情報を共通プロトコルおよびレジストリに従って識別するように構成されている。ロギング制御部が、ロギング済み情報を解析(parse)するように構成され、かつ、ロギング済み情報を解析に従って保存するように構成されている。
【０００５】
本発明のさらなる実施形態において、レジストリは、ロギング・アプリケーションに従って更新可能であるように構成されている。
【０００６】
本発明の別の実施形態において、ロギング・デーモンは、ロギング済み情報を複数のロ

ギング・アプリケーションに従って解析するように構成されている。

【０００７】

本発明のさらなる実施形態において、ロギング・デーモンは、保存されたロギング済み情報の少なくとも一部を複数のロギング・アプリケーションの少なくとも一部の中で共有するように構成されている。

【０００８】

本発明のさらに別の実施形態において、ロギング・エラーを検出した結果として、エラーリソース割当て機能が、ロギング済み情報を保存するための新規のスペースを割り当てるように構成されている。

【発明を実施するための最良の形態】

【０００９】

本発明をより完全に理解するために、添付の図面と合わせて考慮される以下の詳細な説明が参照されるべきである。

【００１０】

本発明を、好ましい実施形態において図を参照して説明する。以下の説明中、同様の符号は同一または類似の要素を表す。本発明は、本発明の目的を達成するための最良の形態に関して説明されるが、これらの教示に照らして本発明の趣旨および範囲から逸脱することなく変更態様を遂行できることが当業者には理解されよう。

【００１１】

図１を参照すると、コンピュータ・システム１００は、少なくとも１つのプログラマブル・コンピュータ・プロセッサからなるあらゆるコンピュータ・システムまたはサブシステムとすることができ、１つまたは複数のアプリケーションを作動し、そこではアプリケーションのトレース、イベント、エラー等がロギングされる。コンピュータ・システム１００は、単一のプログラマブル・コンピュータ・プロセッサおよび接続システムもしくはネットワーク・システムを含むことができ、または、関連のプログラマブル・コンピュータ・プロセッサおよび接続システムもしくはネットワーク・システムのネットワークを含むことができ、あるいは、別のシステムの一部を形成できるサブシステムを含むことができ、これらは全て当業者には知られている。コンピュータ・システム１００の１例として、１つまたは複数のプログラマブル・コンピュータ・プロセッサと、関連の磁気テープ・ライブラリ・システム、例えば磁気テープ・ライブラリおよび或る数の磁気テープ・データ・ストレージ・ドライブを有する仮想テープ・サーバとを含む。コンピュータ・システム１００の別の例は、１つまたは複数のプログラマブル・コンピュータ・プロセッサと、関連の光学ディスク・ライブラリおよび或る数の光学データ・ストレージ・ドライブとを有する仮想光学サーバを含む。この点について、光学とは、ＣＤ（コンパクト・ディスク）、ＤＶＤ（デジタル・バーサタイル・ディスク）、ＨＤ‐ＤＶＤ（高品位ＤＶＤ）、Ｂｌｕ‐Ｒａｙ、およびホログラフィ・ディスクやドライブを含む。コンピュータ・システム１００のさらに別の例は、１つ以上のプログラマブル・コンピュータ・プロセッサと、磁気テープ・ライブラリ・システムをエミュレートする関連のハード・ディスク・ドライブ・アレイとを有する仮想テープ・サーバを含む。

【００１２】

先行技術で知られている複数のロギング・アプリケーション１０２、１０３、１０４は、コンピュータ・システム１００に関する情報、例えばコンピュータ・システムまたはコンピュータ・システムのアプリケーションの操作に関するトレース、イベント、エラー等をロギングするように構成されている。ロギング・アプリケーション１０２、１０３、１０４は、スタンドアロン・ロギング・アプリケーションを含むことができ、または、ロギングの実行もする他のアプリケーションを含むことができる。本発明の１実施形態によれば、ロギング・デーモン１０１が、情報をロギングするためにロギング・アプリケーション１０２、１０３、１０４に対して共通のロギング方法を提供し、単一スレッド（threaded）の「多対多」管理を提供する。ロギング・デーモンは、コンピュータ・システム１００のアプリケーションからなることができ、または、別のロギング・サーバからなること

ができ、または、１つまたは複数のロギング・アプリケーションの一部からなることができる。

【００１３】

ロギング・デーモン１０１は、レジストリおよびロギング制御部を提供するように構成されており、ロギング制御部は、ロギング・アプリケーションに従ってロギング済み情報を解析するように構成されているとともに、ロギング済み情報を解析に従って保存するように構成されている。さらに、エラー処理機能部が、ロギング・アプリケーションに対するエラーまたはロギング・アプリケーションのエラーを処理し、ロギング・エラーを検出した結果として、該ロギング済み情報を保存するための新規のスペースを割り当てるように構成されたエラー・リソース割当て機能部を提供する。

【００１４】

ロギング・デーモン１０１のレジストリは、各ロギング・アプリケーションに、各ロギング・アプリケーション用に個別の識別子を指定するレジストリを提供する。ロギング・アプリケーション１０２、１０３、１０４は、共通のＡＰＩ（アプリケーション・プログラム・インタフェース）呼び出しによってそれらのデータをロギングすることができ、また、それら独自のロギング基準を前もって、または動的に構成することもできる。本発明により、さらに数バイトのデータを元のロギング・データ列に接続し、アプリケーションのレジストリ識別子およびロギング基準を指定することができる。

【００１５】

図４を参照すると、ステップ４０４において、ロギング・アプリケーション（４０２）からロギング・データが受信される。ステップ４０６では、ロギング・デーモンのロギング制御部が、データ・パッケージが有効かどうか（このことは、データ・パッケージが共通ロギング・フォーマットに従って識別されるかどうかを意味し得る）を判断する。有効でない場合、データ・パッケージは「ジャンク」と見なされ、ジャンク・データ・マネージャ４０８へ送信され、放棄される、等。

【００１６】

ステップ４１０において、データ・パッケージが、レジストリに登録されたＩＤを有するかどうかが判断される。有さない場合、データ・パッケージは既存のロギング・アプリケーションへ動的に変化することができ、ステップ４１２においてレジストリから識別子が獲得される。さらに、データ・パッケージは、ステップ４１４で判断されるように、そこにデータが保存されるべき新規のカテゴリを表すことができ、ステップ４１８において新規のリソースが割り当てられ、データが解析されて適切なログへ経路選択される。データ・パッケージが良好であれば、ステップ４１６においてそのリソース宛先に即座に経路選択される。

【００１７】

図６を参照すると、アプリケーションからは新規の通信信号を受信することができる。このアプリケーションは、ステップ６０２において、新規のアプリケーション、または再開された失敗または終了アプリケーションを含むことができる。ロギング・デーモンは、ステップ６０４において、アプリケーションによりロギング・デーモンに第１信号が送信されるとすぐに、アプリケーションとロギング・デーモン自体との間の通信を最適に確立する。ステップ６０６は予め登録されたＩＤが存在するかどうかを判断し、存在すれば、ステップ６１０においてレジストリのテーブルが更新され、存在しなければ、ステップ６０８においてこのアプリケーションが登録される。ステップ６１２は図４のステップ４１２に戻り、アプリケーションのロギング・データフローを正常化してデータ列を適切なデータ・スペースへ経路選択する。別法として、ロギング・デーモンを提供するサーバが失敗または終了することがあるが、ロギング・サーバの不在中に、アプリケーションはなおアクティブにしておくことができる。失敗または終了したサーバが再び復活したとき、例えばステップ６０２～６１２におけるようにアプリケーションのロギング信号が受信されるとすぐに、そのアプリケーションのレジストリのテーブル・エントリが復元される。

【００１８】

図１にデータフローの１例を示す。この図において、ロギング・デーモン１０１のロギング制御部は、それぞれのロギング・アプリケーションにより命令されたリソース宛先に対してアプリケーション１２０、１３０、１４０から着信するデータ・パッケージを解析(parse)し、データ・パッケージをそれぞれのログへ保存する。例えば、アプリケーション＃１（１２０）は、ログ１２１、１２２～１２３に対してデータ・パッケージを解析すべきであることを示すことができ、アプリケーション＃ｎ（１４０）は、ログ１４１、１４２～１４３に対してデータ・パッケージを解析すべきであることを示すことができる。したがって、ロギング制御部は、複数のロギング・アプリケーションに従ってロギング済み情報を解析する。ロギング済み情報の例として、「デバイスは何かまたはアプリケーションは何を実行しているか」、例えばデータ・ストレージ・カートリッジ＃ｘｘｘｘｘをドライブ＃ｘｘｘにマウントしている、データを送信している等に関する情報をロギングするトレース・ログ、特定のイベント、例えばドライブのマイクロコードをｘｘｘｘの時点でアップグレードする、等をロギングするイベント・ログ、およびエラー表示（ソフトまたはハード）を識別するエラー・ログ等を含む。

【００１９】

ロギング・デーモンはロギング済み情報を特定のカテゴリに解析し、これらのカテゴリは、特定のイベント、トレース、またはエラー、あるいは、イベントとエラーとの組み合わせ、あるいはトレースとエラーとの組み合わせ、あるいはトレースとイベントとの組み合わせ、あるいは３つ全ての組み合わせからなるものとすることができる。

【００２０】

別法として、ログの幾つかを同一のデータ・スペースで共有することができる。図２および図５を参照すると、ロギング制御部は、保存されたロギング済み情報の少なくとも一部を複数のロギング・アプリケーションの少なくとも一部の中で共有するように構成されている。ステップ５０４において、ロギング・アプリケーション５０２からロギング・データが受信される。ステップ５０６において、ロギング・デーモンのロギング制御部は、データを共有すべきであるかどうかを判断し、共有すべきでない場合、データはステップ５０８において、アプリケーションの専用スペース、例えば既に検討した図１のログへ経路選択される。ロギング・デーモンのロギング制御部が、データを共有すべきであると判断した場合、ステップ５１０において、データは必要に応じて解析され、次の処置のためにアプリケーションの共有スペースへ経路選択される。図２にデータフローの１例を示す。この図では、ロギング・デーモン１０１のロギング制御部が、それぞれのロギング・アプリケーションにより命令された共有リソース宛先に対して着信するデータ・パッケージを解析し、それぞれのログに対するデータ・パッケージを保存する。例えば、アプリケーション＃１およびアプリケーション＃２は、共有のログ１２１、１２２～１２３に対してデータ・パッケージを解析すべきであることを示すことができる。このように、ロギング制御部は、ロギング済み情報を複数のロギング・アプリケーションに従って解析する。

【００２１】

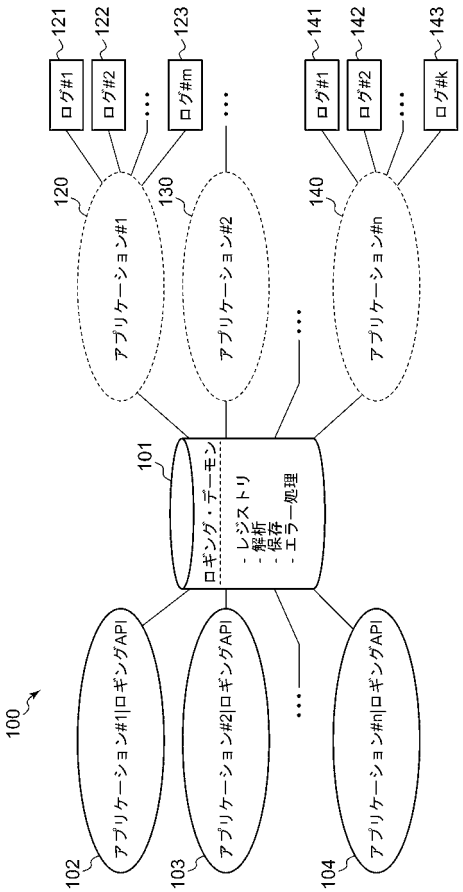
レジストリは、識別子に加え、共有リソースのＩＤおよびデーモン用のメッセージ・キューを維持し、アプリケーションのデータを解析し保存することができる。したがって、図３および図４を参照すると、アプリケーションが止まった場合、ステップ４１０～４１６において、ロギング・デーモン１０１は既にレジストリにある共有のメモリおよびメッセージ・キューを自動的に再確立し、アプリケーション３０１からアプリケーションログ３０２、３０３までデータフローを再確立する。

【００２２】

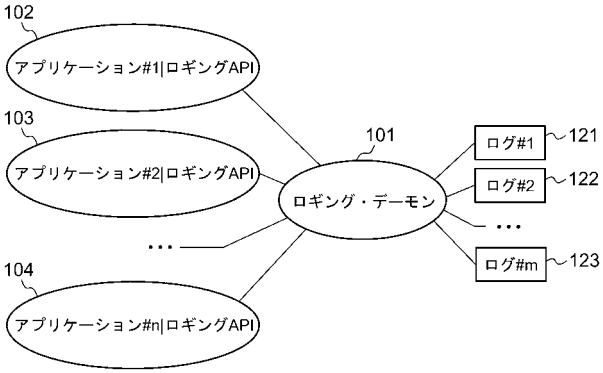
上述したように、ロギング・デーモンを提供するサーバは失敗または終了することがあるが、ロギング・サーバの不在中に、アプリケーションはなおアクティブにしておくことができる。さらに、アプリケーションのロギング・データ・スペースが破損する（別のプログラムにより使用されて）ことがあるが、ロギング・デーモンもアプリケーションもやはり通常通りアクティブである。このような非常処置は、そのロギング・データ列をロギング・デーモンに送信し続けるアプリケーション自体には無効であることがある。

【００２３】
図３および図７を参照すると、ロギング・アプリケーション７０２から、着信するロギング・データ列７０４が受信され、データをロギングしようとする際にエラーが存在する場合（ステップ７０６）、ステップ７０８は、サーバが作動中であるかどうかを判断する。作動中ではない場合、ステップ７１０においてサーバは再起動され（大抵はコンピュータ・システムまたはデーモンが再起動される際）、ステップ７１２においてデータが再送される。さらに、アプリケーションのレジストリのテーブル・エントリが復元され、図６に関して既に検討したように、現在のデータを新規のデータとして扱う。一時バッファがいっぱいである場合（ステップ７１８）、ステップ７２０において一時バッファが空にされ、データが保存される。これらの問題がいずれも問題を引き起こさなかったならば、ステップ７１４において再試行を実行することができ、ステップ７１６においてエラー情報が保存される。
【００２４】
ステップ７２２は、ロギング済みデータが保存されたリソースが良好であるかどうかを判断し、良好でない場合、ステップ７２４においてリソース３０２は削除され、ステップ７２８において新規のリソース３０３が割り当てられる。ステップ７２６は、新規のスペースが必要であるかどうかを判断し、必要である場合、ステップ７２８において、付加されたリソース３０３が割り当てられ、ステップ７３０において次の処置が講じられる。
【００２５】
図１、図２、および図３のデータフローに関しては変更することができるということ、ならびにステップを並び替え、図４、図５、図６、および図７のフローダイヤグラムに別の変更を加えることができるということを当業者ならば理解するであろう。さらに、本明細書で検討する配置とは異なる特定の要素の配置を採用することができるということを当業者ならば理解するであろう。
【００２６】
本発明の好ましい実施形態を詳細に説明したが、これらの実施形態への修正および改作を、以下の特許請求の範囲で説明するような本発明の範囲から逸脱することなく当業者は思いつくことができるということが明らかとなるべきである。
【図面の簡単な説明】
【００２７】
【図１】本発明を実施するコンピュータ・システムおよびロギング・データフローの１実施形態の概略図である。
【図２】スペース共有を示す１実施形態における、図１のコンピュータ・システムおよびロギング・データフローの概略図である。
【図３】ロギング・データフローのリカバリを示す１実施形態における、図１のコンピュータ・システムおよびロギング・データフローの概略図である。
【図４】図１のコンピュータ・システムおよびロギング・データフローのレジストリの１実施形態を示すフローチャートである。
【図５】図２のコンピュータ・システムおよびロギング・データフローのスペース共有の１実施形態を示すフローチャートである。
【図６】図３のコンピュータ・システムおよびロギング・データフローのリカバリ操作の１実施形態を示すフローチャートである。
【図７】図３のコンピュータ・システムおよびロギング・データフローのリカバリ操作の１実施形態を示すフローチャートである。

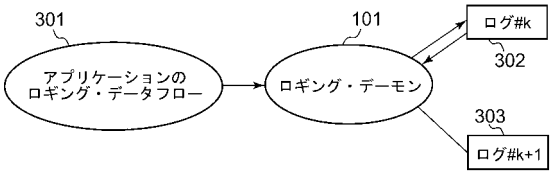
【図１】


100
101
ロギング・デーモン
- レジストリ
- 解析
- 保存
- エラー処理
120
アプリケーション#1
130
アプリケーション#2
140
アプリケーション#n
121
ログ#1
122
ログ#2
123
ログ#m
141
ログ#1
142
ログ#2
143
ログ#k
102
アプリケーション#1|ロギングAPI
103
アプリケーション#2|ロギングAPI
104
アプリケーション#n|ロギングAPI

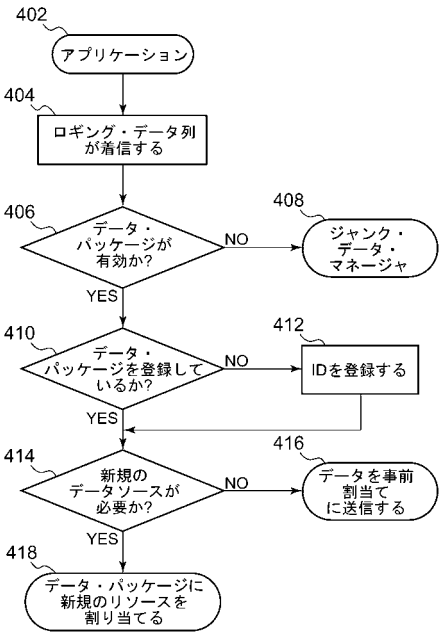
【図２】


102
アプリケーション#1|ロギングAPI
103
アプリケーション#2|ロギングAPI
104
アプリケーション#n|ロギングAPI
101
ロギング・デーモン
ログ#1
121
ログ#2
122
ログ#m
123

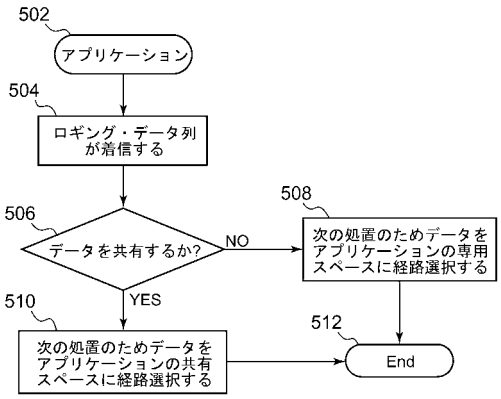
【図３】


301
アプリケーションの
ロギング・データフロー
101
ロギング・デーモン
ログ#k
302
303
ログ#k+1

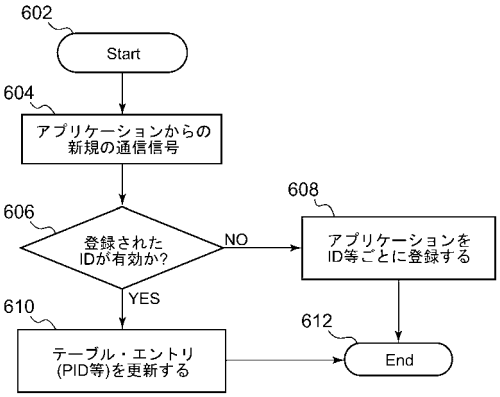
【図４】


402
アプリケーション
404
ロギング・データ列
が着信する
406
データ・
パッケージが
有効か?
NO
408
ジャンク・
データ・
マネージャ
YES
410
データ・
パッケージを登録して
いるか?
NO
412
IDを登録する
YES
414
新規の
データソースが
必要か?
NO
416
データを事前
割当て
に送信する
YES
418
データ・パッケージに
新規のリソースを
割り当てる

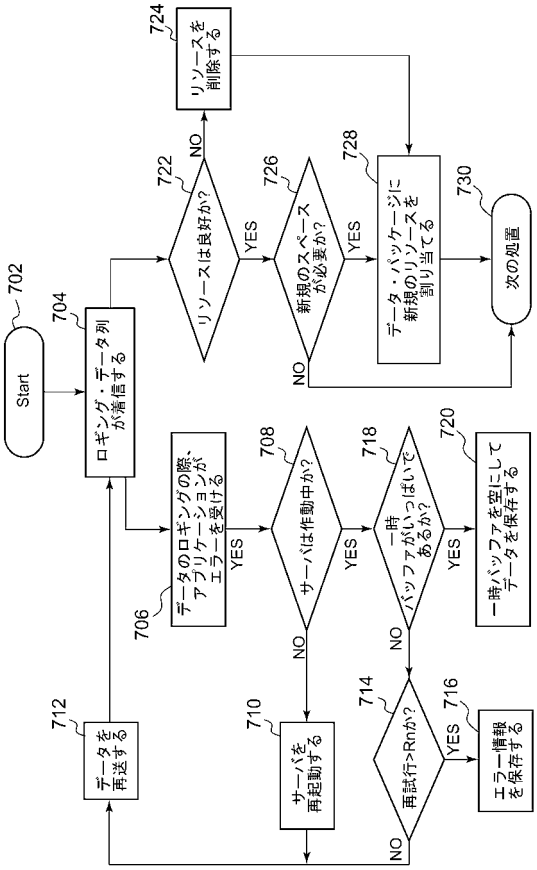
【図５】


502
アプリケーション
504
ロギング・データ列
が着信する
506
データを共有するか?
NO
508
次の処置のためデータを
アプリケーションの専用
スペースに経路選択する
YES
510
次の処置のためデータを
アプリケーションの共有
スペースに経路選択する
512
End

【図６】


602
Start
604
アプリケーションからの
新規の通信信号
606
登録された
IDが有効か?
NO
608
アプリケーションを
ID等ごとに登録する
YES
610
テーブル・エントリ
(PID等)を更新する
612
End

【図７】


Start
702
ロギング・データ列
が着信する
704
データのロギングの際、
アプリケーションが
エラーを受ける
706
YES
サーバは作動中か?
708
NO
サーバを
再起動する
710
YES
一時
バッファがいっぱいで
あるか?
718
NO
再試行>Rnか?
714
YES
エラー情報
を保存する
716
NO
データを
再送する
712
YES
一時バッファを空にして
データを保存する
720
リソースは良好か?
722
NO
リソースを
削除する
724
YES
新規のスペース
が必要か?
726
YES
データ・パッケージに
新規のリソースを
割り当てる
728
NO
次の処置
730

フロントページの続き

(74)代理人 100086243
弁理士 坂口 博
(72)発明者 ダニエル・ジェイムズ・ワイナルスキー
アメリカ合衆国８５７１０－６２３７ アリゾナ州ツーソン サウス・ウッドストック・ドライヴ６４７
(72)発明者 ヘンリー・ジャン・リュー
アメリカ合衆国８５７１５ アリゾナ州ツーソン ノース・サンドストーン・リッジ・ドライヴ１７７０
(72)発明者 ラルフ・トーマス・ビーストン
アメリカ合衆国８５７４７－９７０７ アリゾナ州ツーソン サウス・アヴェニーダ・デ・ベレーザ７６４０
(72)発明者 トーマス・ウィリアム・ビッシュ
アメリカ合衆国８５７４９ アリゾナ州ツーソン エヌ・フェニモア・アヴェニュー２０３８

Ｆターム(参考) 5B042 MA08 MA14 MC40