Title: ISOLATING, MANAGING AND COMMUNICATING WITH USER INTERFACE ELEMENTS

Abstract: Various technologies and techniques are disclosed that allow a user interface in a managed runtime environment to display content from two or more different partitioned process spaces. A software application having a user interface operates in a managed runtime environment. A window handle allows a user interface of the application to display content from different partitioned process spaces at a time. Information from the partitioned process spaces can be integrated within the user interface while preserving a level of isolation between the partitioned process spaces. This is accomplished by providing a first portion of a content container for the user interface with a host control that contains the window handle and resides in a first partitioned process space of a main window. A second portion of the content container is provided with a source control that provides the window handle and resides in a second partitioned process space.
ISOLATING, MANAGING AND COMMUNICATING
WITH USER INTERFACE ELEMENTS

BACKGROUND

[001] In modern software development, many applications are written to execute within a managed runtime environment, such as the MICROSOFT® .NET Framework. Managed runtime environments serve as a way to allow applications access to system resources, enforce security policies, etc. A managed runtime environment accomplishes this by creating "partitioned process spaces" wherein an application is executed. A partitioned process space allows an application to execute in such a manner that it is isolated from any other partitioned spaces in the process that the computer may be executing at the time. In the MICROSOFT® .NET Framework, these partitioned process spaces are called "AppDomains."

[002] A problem arises when an application is executed in one partitioned process space and the content is required to execute in another partitioned process space. Given the isolation and other rules governing managed runtime environments discussed earlier, the application will not be able to communicate with the content in the different partitioned process space, nor will the content be shown as part of the application.

SUMMARY

[003] Various technologies and techniques are disclosed that allow a user interface in a managed runtime environment to display content from two or more different partitioned process spaces. A particular software application is provided that operates in a managed runtime environment, the particular software application having a user interface. A window handle is used to allow a user interface of the particular software application to display content from two or more different partitioned process spaces at a same time. Information from the two or more partitioned process spaces can be integrated within the user interface of the software application while preserving a level of isolation between the two or more different partitioned process spaces. This is accomplished by providing a first portion of a content container for the user interface with a host control that will contain a provided window as a child window and resides in a first partitioned
process space of a main window. A second portion of the content container for the user interface is provided with a source control that provides the window handle and resides in a second partitioned process space of the main window. In one implementation, the second portion of the content container can have content for display, such as in an isolated frame control.

[004] This Summary was provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

**BRIEF DESCRIPTION OF THE DRAWINGS**

[005] Figure 1 is a diagrammatic view of a computer system of one implementation.

[006] Figure 2 is a diagrammatic view of a managed software application of one implementation operating on the computer system of Figure 1.

[007] Figure 3 is a process flow diagram illustrating the high level stages involved in one implementation of the system of Figure 1.

[008] Figure 4 is a process flow diagram for one implementation of the system of Figure 1 illustrating the stages involved in allowing content from different partitioned process spaces to be displayed together and to communicate with each other.

[009] Figure 5 is a process flow diagram for one implementation of the high level stages of the system of Figure 1 illustrating the stages involved in allowing content from different partitioned process spaces to be displayed together with different levels of security.

[010] Figure 6 illustrates a logical diagram of one implementation showing external content appearing within an isolated frame control.

[011] Figure 7 is a logical diagram for one implementation illustrates using a window handle to allow content from different partitioned process spaces to be displayed in the same user interface.
DETAILED DESCRIPTION

[012] For the purposes of promoting an understanding of the principles of the invention, reference will now be made to the embodiments illustrated in the drawings and specific language will be used to describe the same. It will nevertheless be understood that no limitation of the scope is thereby intended. Any alterations and further modifications in the described embodiments, and any further applications of the principles as described herein are contemplated as would normally occur to one skilled in the art.

[013] The system may be described in the general context as an application that allows content from different partitioned process spaces to be displayed in the same user interface in a managed environment, but the system also serves other purposes in addition to these. In one implementation, one or more of the techniques described herein can be implemented as features within a framework environment such as MICROSOFT® .NET Framework, or from any other type of program or service that allows for execution and/or development of software.

[014] As shown in Figure 1, an exemplary computer system to use for implementing one or more parts of the system includes a computing device, such as computing device 100. In its most basic configuration, computing device 100 typically includes at least one processing unit 102 and memory 104. Depending on the exact configuration and type of computing device, memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. This most basic configuration is illustrated in Figure 1 by dashed line 106.

[015] Additionally, device 100 may also have additional features/functionality. For example, device 100 may also include additional storage (removable and/or non-removable) including, but not limited to, magnetic or optical disks or tape. Such additional storage is illustrated in Figure 1 by removable storage 108 and non-removable storage 110. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Memory 104, removable storage 108 and non-
removable storage 110 are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by device 100. Any such computer storage media may be part of device 100.

[016] Computing device 100 includes one or more communication connections 114 that allow computing device 100 to communicate with other computers/applications 115. Device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 111 such as a display, speakers, printer, etc. may also be included. These devices are well known in the art and need not be discussed at length here. In one implementation, computing device 100 includes managed software application 200.

Managed software application 200 will be described in further detail in Figure 2.

[017] Turning now to Figure 2 with continued reference to Figure 1, a managed software application 200 operating on computing device 100 is illustrated. Managed software application 200 is one of the application programs that reside on computing device 100. However, it will be understood that managed software application 200 can alternatively or additionally be embodied as computer-executable instructions on one or more computers and/or in different variations than shown on Figure 1. Alternatively or additionally, one or more parts of managed software application 200 can be part of system memory 104, on other computers and/or applications 115, or other such variations as would occur to one in the computer software art.

[018] Managed software application 200 includes program logic 204, which is responsible for carrying out some or all of the techniques described herein. Program logic 204 includes logic for providing a particular software application that operates in a managed runtime environment (.NET Framework, etc.) and has a user interface 206; logic for using a window handle (e.g. MICROSOFT® Windows HWND handle) to allow a user interface of the particular software
application to display content from two or more different partitioned process spaces (e.g. AppDomains) at a same time 208; logic for allowing information from the two or more partitioned process spaces to be integrated within the user interface of the software application operating in the managed runtime environment while preserving a level of isolation between the two or more different partitioned process spaces 210; logic for enabling a first one of the partitioned process spaces to communicate with a second one of the partitioned process spaces by performing a marshal by reference process (MarshalByRefObjects in the windows operating system) 212; logic for enabling a first one of the partitioned process spaces to have a first portion of content container with a host control that contains a provided window handle, and a second one of the partitioned process spaces with a second portion of the content container with a source control that provides the window handle 214; logic for allowing a first security level that the first portion of the content container follows to be different than a second security level than the second portion of the content container follows (e.g. the second being from an entrusted or different source, etc.) 216; and other logic for operating the application 220. In one implementation, program logic 204 is operable to be called programmatically from another program, such as using a single call to a procedure in program logic 204.

[019] Turning now to Figures 3-5 with continued reference to Figures 1-2, the stages for implementing one or more implementations of managed software application 200 are described in further detail. Figure 3 is a process flow diagram illustrating the high level stages involved in one implementation of the system of Figure 1. In one form, the process of Figure 3 is at least partially implemented in the operating logic of computing device 100. The process begins at start point 250 with providing a particular software application that operates in a managed runtime environment (e.g. .NET Framework, etc.), the particular software application having a user interface (stage 252). A first portion of a content container for the user interface has a host control that contains a provided window handle (e.g. MICROSOFT® Windows HWND handle) and resides in a first partitioned process space (e.g. AppDomain, etc.) of a main window (stage 254). A second portion of
the content container for the user interface is provided with a source control that provides the window handle and resides in a second partitioned process space of the main window, the second portion having content for display (stage 256). Information from the first partitioned process space and the second partitioned process space can be integrated within the user interface while preserving a level of isolation between the first portion and the second portion of the content container (stage 258). Different security levels can optionally be provided/enforced for the first and second partitioned process spaces (e.g. such as when one is from an untrusted or unknown source, etc.) (stage 260). The process ends at end point 262.

Figure 4 is a process flow diagram for one implementation of the system of Figure 1 illustrating the stages involved in allowing content from different partitioned process spaces to be displayed together and to communicate with each other. In one form, the process of Figure 4 is at least partially implemented in the operating logic of computing device 100. The process begins at start point 270 with providing a particular software application that operates in a managed .NET framework environment, the particular software application having a user interface (stage 272). A first portion of a content container is provided for the user interface with a host control that contains a MICROSOFT® Windows HWND handle and resides in a first partitioned process space (e.g. AppDomain, etc.) for a main window (stage 274). A second portion of the content container is provided for the user interface with a source control that provides the MICROSOFT® Windows HWND handle and resides in a second partitioned process space of the main window (stage 276). In one implementation, the second portion of the content container has content for display, such as in an isolated frame control (stage 276). The first partitioned process space can communicate with a second partitioned process space by performing a MarshalByRefObjects process (stage 278). The process ends at end point 280.

Figure 5 is a process flow diagram for one implementation of the high level stages of the system of Figure 1 illustrating the stages involved in allowing content from different partitioned process spaces to be displayed together with different levels of security. In one form, the process of Figure 5 is at least partially
implemented in the operating logic of computing device 100. The process begins at start point 290 with providing a particular software application that operates in a managed framework environment (stage 292). A user interface of the particular software application can have content areas that come from multiple AppDomains by using MICROSOFT® Windows HWND interop (stage 294). Visual elements can be merged into the same user interface, while maintaining isolation and management functions provided by the AppDomains, while allowing for two-way communications between the AppDomains using marshaling by reference (stage 296). Optionally, a user, such as the software developer, can declare security levels to use and/or other details about one or more of the content areas (such as in an XAML file, or in various other formats) (stage 298). This allows the content areas to be displayed with varying levels of security authority as desired (stage 298). For example, it may be desirable to vary the security levels depending on whether the content is from a trusted or untrusted source, etc. (stage 298). The process ends at end point 299.

[022] Figure 6 is a logical diagram for one implementation that illustrates external content appearing within an isolated frame control. Application main window 300 is some graphical application that contains window contents 305. Window contents 305 can be any kind of visual element that is displayed to the user. Application main window 300 also contains isolated frame control 310. Isolated frame control 310 will contain content 320 where content 320 is external to the overall application. Isolated frame control 310 may be invisible to the user. In one implementation, some or all of the techniques discussed herein are used to allow content 320 (such as from a separate entity that is untrusted) to be displayed in an isolated frame control that operates in a different partitioned space than the window contents 305.

[023] Figure 7 is a logical diagram for one implementation illustrates using a window handle to allow content from different partitioned process spaces to be displayed in the same user interface. Process 350 will create a first partitioned process space 358 within a managed runtime environment 352. Within the partitioned process space 358, an application 360 is executed. The application 360
contains an isolated frame control 362 wherein there is a handle source 366. The isolated frame control will be loading content 364 so that it is visible to a user. Content 364 is external to application 360 and, as such, will require the process 350 to create a second partitioned process space 356 wherein the content 364 shall be executed. When the content 364 is loaded, one of its components, a handle host 368 registers itself with a handle provider 354 as a window handle 370. The handle source 366 can then access the window handle 370 and show the content 364 within the isolated frame control 362 contained within the application 360.

[024] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims. All equivalents, changes, and modifications that come within the spirit of the implementations as described herein and/or by the following claims are desired to be protected.

[025] For example, a person of ordinary skill in the computer software art will recognize that the client and/or server arrangements, user interface screen content, and/or data layouts as described in the examples discussed herein could be organized differently on one or more computers to include fewer or additional options or features than as portrayed in the examples.
What is claimed is:

1. A computer-readable medium having computer-executable instructions for causing a computer to perform steps comprising:
   - provide a particular software application that operates in a managed runtime environment, the particular software application having a user interface;
   - use a window handle to allow a user interface of the particular software application to display content from two or more different partitioned process spaces at a same time; and
   - allow information from the two or more partitioned process spaces to be integrated within the user interface of the software application operating in the managed runtime environment while preserving a level of isolation between the two or more different partitioned process spaces.

2. The computer-readable medium of claim 1, wherein the window handle is a Windows HWND handle.

3. The computer-readable medium of claim 1, wherein the managed runtime environment is a managed .NET Framework environment.

4. The computer-readable medium of claim 1, wherein the one or more partitioned process spaces are AppDomains in the managed runtime environment.

5. The computer-readable medium of claim 1, wherein a first one of the partitioned process spaces is operable to communicate with a second one of the partitioned process spaces by performing a marshal by reference process.

6. The computer-readable medium of claim 5, wherein the marshal by reference process uses a MarshalByRefObjects process in a Windows operating system.

7. The computer-readable medium of claim 1, wherein a first one of the partitioned process spaces has a first portion of a content container for the user interface with a host control that contains a window handle; and
   - wherein a second one of the partitioned process spaces has a second portion of the content container for the user interface with a source control that provides the window handle.
8. The computer-readable medium of claim 7, wherein a first security level that the first portion of the content container follows is different than a second security level that the second portion of the content container follows.

9. The computer-readable medium of claim 8, wherein the second portion of the content container displays content from an untrusted source.

10. A method for integrating user interface content from different partitioned process spaces in a single user interface comprising the steps of:

   providing a particular software application that operates in a managed runtime environment, the particular software application having a user interface;

   providing a first portion of a content container for the user interface with a host control that contains a window handle and resides in a first partitioned process space of a main window;

   providing a second portion of the content container for the user interface with a source control that provides the window handle and resides in a second partitioned process space of the main window, the second portion of the content container having content for display; and

   allowing information from the first partitioned process space and the second partitioned process space to be integrated within the user interface of the software application operating in the managed runtime environment while preserving a level of isolation between the first portion and the second portion of the content container.

11. The method of claim 10, wherein the managed runtime environment is a managed .NET Framework environment.

12. The method of claim 10, wherein the first partitioned process space and the second partitioned process space are AppDomains in the managed runtime environment.

13. The method of claim 10, wherein the window handle is a Windows HWND handle.

14. The method of claim 10, wherein a first security level that the first portion of the content container follows is different than a second security level that the second portion of the content container follows.
15. A computer-readable medium having computer-executable instructions for causing a computer to perform the steps recited in claim 10.

16. A method for integrating user interface content from different partitioned process spaces in a single user interface comprising the steps of:

- providing a particular software application that operates in a managed .NET Framework environment, the particular software application having a user interface;
- providing a first portion of a content container for the user interface with a host control that contains a Windows HWND handle and resides in a first partitioned process space of a main window; and
- providing a second portion of the content container for the user interface with a source control that provides the Windows HWND handle and resides in a second partitioned process space of the main window, the second portion of the content container having content for display.

17. The method of claim 16, wherein by using the Windows HWND handle from the first portion of the content container and from the second portion of the content container, allowing information from more than one partitioned process space to be integrated within the user interface of the software application operating in the managed .NET Framework environment while preserving a level of isolation between the first portion and the second portion.

18. The method of claim 16, wherein the first partitioned process space and the second partitioned process space are AppDomains in the managed .NET Framework environment.

19. The method of claim 16, wherein the first partitioned process space is operable to communicate with a second partitioned process space by performing a MarshalByRefObjects process.

20. A computer-readable medium having computer-executable instructions for causing a computer to perform the steps recited in claim 16.
<table>
<thead>
<tr>
<th>MANAGED SOFTWARE APPLICATION (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM LOGIC (204)</td>
</tr>
<tr>
<td>LOGIC FOR PROVIDING A PARTICULAR SOFTWARE APPLICATION THAT OPERATES IN A MANAGED RUNTIME ENVIRONMENT (e.g., .NET FRAMEWORK, ETC.) AND HAS A USER INTERFACE (206)</td>
</tr>
<tr>
<td>LOGIC FOR USING A WINDOW HANDLE (e.g., WINDOWS HWND HANDLE) TO ALLOW A USER INTERFACE OF THE PARTICULAR SOFTWARE APPLICATION TO DISPLAY CONTENT FROM TWO OR MORE DIFFERENT PARTITIONED PROCESS SPACES (e.g., APPDOMAINS) AT A SAME TIME (208)</td>
</tr>
<tr>
<td>LOGIC FOR ALLOWING INFORMATION FROM THE TWO OR MORE PARTITIONED PROCESS SPACES TO BE INTEGRATED WITHIN THE USER INTERFACE OF THE SOFTWARE APPLICATION OPERATING IN THE MANAGED RUNTIME ENVIRONMENT WHILE PRESERVING A LEVEL OF ISOLATION BETWEEN THE TWO OR MORE DIFFERENT PARTITIONED PROCESS SPACES (210)</td>
</tr>
<tr>
<td>LOGIC FOR ENABLING A FIRST ONE OF THE PARTITIONED PROCESS SPACES TO COMMUNICATE WITH A SECOND ONE OF THE PARTITIONED PROCESS SPACES BY PERFORMING A MARSHEL BY REFERENCE PROCESS (MARSHELBYREFOBJECTS IN THE WINDOWS OPERATING SYSTEM) (212)</td>
</tr>
<tr>
<td>LOGIC FOR ENABLING A FIRST ONE OF THE PARTITIONED PROCESS SPACES TO HAVE A FIRST PORTION OF A CONTENT CONTAINER WITH A HOST CONTROL THAT CONTAINS A PROVIDED WINDOW HANDLE, AND A SECOND ONE OF THE PARTITIONED PROCESS SPACES WITH A SECOND PORTION OF THE CONTENT CONTAINER WITH A SOURCE CONTROL THAT PROVIDES THE WINDOW HANDLE (214)</td>
</tr>
<tr>
<td>LOGIC FOR ALLOWING A FIRST SECURITY LEVEL THAT THE FIRST PORTION OF THE CONTENT CONTAINER FOLLOWS TO BE DIFFERENT THAN A SECOND SECURITY LEVEL THAN THE SECOND PORTION OF THE CONTENT CONTAINER FOLLOWS (E.g., THE SECOND BEING FROM AN UNTRusted OR DIFFERENT SOURCE, ETC.) (216)</td>
</tr>
<tr>
<td>OTHER LOGIC FOR OPERATING THE APPLICATION (220)</td>
</tr>
</tbody>
</table>
START 250

PROVIDE A PARTICULAR SOFTWARE APPLICATION THAT OPERATES IN A MANAGED RUNTIME ENVIRONMENT (E.G. .NET FRAMEWORK, ETC.), THE PARTICULAR SOFTWARE APPLICATION HAVING A USER INTERFACE 252

PROVIDE A FIRST PORTION OF A CONTENT CONTAINER FOR THE USER INTERFACE WITH A HOST CONTROL THAT.Contains A PROVIDED WINDOW HANDLE (E.G. WINDOWS HWND HANDLE) AND RESIDES IN A FIRST PARTITIONED PROCESS SPACE (E.G. APPDOMAIN, ETC.) OF A MAIN WINDOW 254


OPTIONALLY ALLOW DIFFERENT SECURITY LEVELS TO BE PROVIDED/ENFORCED FOR THE FIRST AND SECOND PARTITIONED PROCESS SPACES (E.G. SUCH AS WHEN ONE IS FROM AN UNTRUSTED OR UNKNOWN SOURCE, ETC.) 260

FIG. 3

END 262
START 270

PROVIDE A PARTICULAR SOFTWARE APPLICATION THAT OPERATES IN A MANAGED .NET FRAMEWORK ENVIRONMENT, THE PARTICULAR SOFTWARE APPLICATION HAVING A USER INTERFACE 272

PROVIDE A FIRST PORTION OF A CONTENT CONTAINER FOR THE USER INTERFACE WITH A HOST CONTROL THAT CONTAINS A WINDOWS HWND HANDLE AND RESIDES IN A FIRST PARTITIONED PROCESS SPACE (E.G. APPDOMAIN, ETC.) OF A MAIN WINDOW 274


ENABLE THE FIRST PARTITIONED PROCESS SPACE TO COMMUNICATE WITH A SECOND PARTITIONED PROCESS SPACE BY PERFORMING A MARSHALBYREFOBJECTS PROCESS 278

END 280

FIG. 4
START 290

PROVIDE A PARTICULAR SOFTWARE APPLICATION THAT OPERATES IN A MANAGED FRAMEWORK ENVIRONMENT 292

ALLOW A USER INTERFACE OF THE PARTICULAR SOFTWARE APPLICATION TO HAVE CONTENT AREAS THAT COME FROM MULTIPLE APPDOMAINS BY USING HWND INTEROP 294

MERGE VISUAL ELEMENTS INTO THE SAME USER INTERFACE, WHILE MAINTAINING ISOLATION AND MANAGEMENT FUNCTIONS PROVIDED BY THE APPDOMAINS, WHILE ALLOWING FOR TWO-WAY COMMUNICATIONS BETWEEN THE APPDOMAINS USING MARSHALING BY REFERENCE 296

OPTIONALLY ALLOW A USER TO DECLARE SECURITY LEVELS TO USE AND/OR OTHER DETAILS ABOUT ONE OR MORE OF THE CONTENT AREAS (SUCH AS IN AN XAML FILE, OR IN VARIOUS OTHER FORMATS), TO ALLOW THE CONTENT AREAS TO BE DISPLAYED WITH VARYING LEVELS OF SECURITY AUTHORITY AS DESIRED (E.G. DEPENDING ON WHETHER THEY ARE FROM A TRUSTED OR UNTRUSTED SOURCE, ETC.) 298

END 299

FIG. 5
A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/00(2006.01)1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 8 G06F 17/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models since 1975
Japanese utility models and applications for utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(KIPO internal) "managed runtime environment", "MRTE", "process space", "window handle", "HWND", "NET", "isolation", "information", "user interface"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 6,453,460 B1 (KEYES, C K) 17 September 2002 See abstract, figures 3-4, claim 16, column 8, line 42 - column 8, line 67</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 2004-0237071 A1 (HOLLANDER, Y et al) 25 November 2004 See abstract, figure 1, claims 1 and 7</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 6,161,148 A (PRATT, J M et al) 12 December 2000 See abstract, figure IA and IB, claim 12</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 6,545,691 B1 (VALLEJO, D M) 8 April 2003 See abstract, figure 2, claims 1 and 18</td>
<td>1-20</td>
</tr>
</tbody>
</table>

See patent family annex

Date of the actual completion of the international search
14 JULY 2008 (14 07 2008)

Date of mailing of the international search report
14 JULY 2008 (14.07.2008)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seogu, Daejeon 302-701, Republic of Korea

Facsimile No 82-42-472-7140

Authorized officer
Lee Hyoung Il

Telephone No 82-42-481-5655
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>JP 2000-32226 1 A2</td>
<td>24.11.2000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6453460 BA</td>
<td>17.09.2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 200 1-15986 A5</td>
<td>30.05.2001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2002-48883 A1</td>
<td>05.12.2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2002-48883 A5</td>
<td>05.12.2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2002-48883 B2</td>
<td>05.12.2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 768758 B2</td>
<td>08.01.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2386 100 AA</td>
<td>13.11.2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2004-023707 1 A1</td>
<td>25.11.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2004-23707 1 AA</td>
<td>25.11.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2005-177752 A1</td>
<td>11.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2005-177752 AA</td>
<td>11.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6823460 B1</td>
<td>23.11.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6823460 BA</td>
<td>23.11.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 72 13 153 BB</td>
<td>01.05.2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 728 126 BB</td>
<td>09.10.2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 200 1-37095 A1</td>
<td>25.05.2001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 10-133878 A2</td>
<td>22.05.1998</td>
</tr>
<tr>
<td>US 654569 1 B1</td>
<td>08.04.2003</td>
<td>US 6545691 B1</td>
<td>08.04.2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6545691 BA</td>
<td>08.04.2003</td>
</tr>
</tbody>
</table>

Form PCT/ISA/210 (patent family annex) (July 2008)