J Extra Corpor Technol. 2018;50:102–12
The Journal of ExtraCorporeal Technology

Integration of Electronic Perfusion Data for Perfusion Registries

Richard F. Newland, BSc, CCP (Aust); Robert A. Baker, PhD, CCP (Aust); Chris A. Barratt, B AppSc(Comp)

Flinders Medical Centre and Flinders University, Bedford Park, Adelaide, South Australia, Australia

Abstract: Although the potential for the utilization of electronic perfusion data (EPD) from proprietary software to facilitate the understanding and improvement of cardiopulmonary bypass (CPB) has been recognized, the generalizability of previous reports of EPD integration are limited by superceded software or lack of sufficient detail for reproducibility. To date, the Australian and New Zealand Collaborative Perfusion Registry (ANZCPR) is the only multicentre perfusion registry to have reported the integration of EPD. The inclusion of EPD in analyses of the impact of CPB on patient outcome is important in improving the understanding of CPB practice. Perfusion registries play an important role in this process, and the incorporation of EPD into perfusion registries could make a significant contribution toward this objective. By sharing the methodology used to integrate EPD from the CONNECT™ software into the ANZCPR, our intent is to diminish some of the barriers to adoption of EPD integration into other perfusion registries, by providing an example of how EPD integration may be achieved. **Keywords:** cardiopulmonary bypass, database, registry, quality improvement. *J Extra Corpor Technol. 2018;50:102–12*

The potential for the utilization of electronic perfusion data (EPD) from proprietary software to facilitate the understanding and improvement of cardiopulmonary bypass (CPB) has been recognized (1–3). We have previously reported how the integration of EPD into perfusion registries can facilitate quality improvement through benchmarking (4). To date, the Australian and New Zealand Collaborative Perfusion Registry (ANZCPR) is the only multicenter perfusion registry to have reported the integration of EPD. We reported a method for the automation of feedback of CPB quality parameters using the Data Management System (DMS) software (Stockert, Munich, Germany) (1); however, upgrades to this software have ceased and it has been replaced by CONNECT™ (LivaNova PLC, London, UK). Furthermore, our previous work did not describe in sufficient detail the methodology used to transfer data from the DMS to our local registry. These limitations may be contributing factors for the nonintegration of EPD into other registries. This article describes, in detail, the methods used to integrate EPD from the CONNECT™ software into the ANZCPR.

Received for publication September 26, 2017; accepted January 29, 2018.
Address correspondence to: Richard F. Newland, BSc, CCP (Aust), Cardiac & Thoracic Surgical Unit, Level 6 Flinders Private Hospital, Bedford Park, Adelaide 5042, South Australia, Australia. E-mail: richard.newland@sa.gov.au
The senior author has stated that the authors have reported no material, financial, or other relationship with any healthcare-related business or other entity whose products or services are discussed in this article.

DESCRIPTION

Database Structure and Connection

Microsoft Access (MS Access) (Microsoft Corporation, Redmond, WA) was chosen as the database platform to integrate EPD based on the availability of the software at participating ANZCPR hospitals and our experience using MS Access in the integration of EPD in our own institution. CONNECT™ uses a Microsoft Structured Query Language (SQL) Server database platform, which can be accessed via the Microsoft SQL Server Management Studio application. CONNECT™ is installed as two separate programs, namely, the Manager program, which has access to all patient records, and the Recorder program, which is used on the heart–lung machine for collection of individual patient data during CPB. An open database connection (ODBC) was used to create a connection between MS Access and the CONNECT™ Manager database. The ANZCPR database structure consists of a Server database,

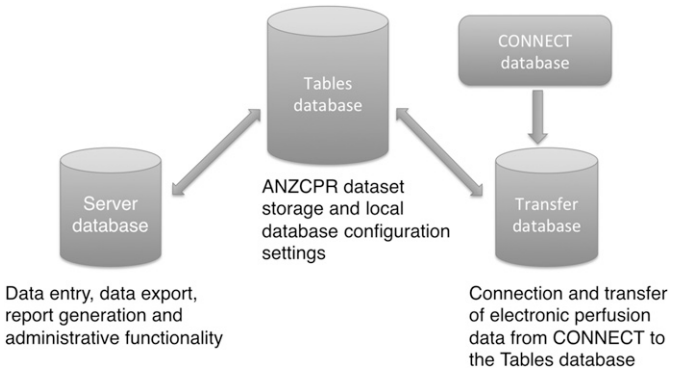

Figure 1. ANZCPR database structure.

a Tables database and a Transfer database (Figure 1). The clinical data-set for the ANZCPR is stored in the Tables database. The Server database provides a front end to the data-set and provides data entry, data export, report generation, and administrative functionality. The Transfer database provides connection and transfer of EPD from CONNECT™ to the Tables database. Each participating ANZCPR hospital has one Tables and Transfer database, but may have multiple Server databases. In a local area network environment, this design allows access to the ANZCPR from either desktop or heart–lung machine computers by linking each instance of the Server database to the Tables and Transfer databases as linked MS Access tables. Using Microsoft Windows 7, ODBC connections were created using the ODBC Data Source Administrator window from the Administrative Tools tab to link the CONNECT™ tables to the Transfer database.

Administrative rights may be required to achieve integration of EPD from CONNECT™ or perform other tasks as outlined in this document; therefore, consultation and assistance from Hospital IT departments are recommended.

CONNECT™ may either be installed on a local computer or on a hospital-based SQL server. During installation the authentication method is set as either integrated Windows authentication or SQL server authentication. Using the ODBC Data Source Administrator window, a connection to CONNECT™ is configured using the SQL Server Native Client 10.0 driver if the CONNECT™ database is installed on the local computer, or using the SQL Server driver if it is installed on the hospital SQL server. To add a new data source to a locally installed CONNECT™ database, the server name is localhost\sqlexpress. For SQL server connection, the hospital server name is required. Authentication using integrated Windows authentication or SQL server authentication is selected as appropriate. The default database is set to "ConnectManager." An ODBC connection is required for each computer on which the Server database is used for EPD integration.

Linking Patient Records, Data Encryption and De-Identification of Patient Data

Patient data records in ANZCPR are linked to the corresponding patient data in CONNECT™ by entering the ANZCPR CPB procedure number into the 'Case record' field in CONNECT™. Each participating ANZCPR hospital may choose their own CPB procedure numbering system. Within the ANZCPR databases, the CPB procedure number is used as the primary key to link data between tables. Patient identifying data is transferred only to the participating ANZCPR hospital database, and are not exported as part of the collaborative ANZCPR data harvest as per the ethics requirements of the ANZCPR (386.15, Southern Adelaide Clinical Human Research Ethics Committee). Within the ANZCPR Server database, the "demography" table is used to generate and store an autonumber that serves as the unique registry ID for each procedure. A separate table stores a unique hospital ID value. When data is exported for the collaborative data harvest, the hospital ID and the registry ID are exported to create a combination of unique record identifiers without patient identification. CONNECT™ has the option of encrypting the patient identifying data (default) which is set as default in the CONNECT™ Manager configuration file. It is not possible to link records in CONNECT™ for integration of EPD without decryption of patient identification. Instructions for editing the configuration file can be found in the CONNECT™ service manual [(5), Section 4.2, p. 62]. In the following command line, value="true" is changed to value="false" to enable decryption and linkage of patient data;

Encrypted: <add key="Database.PatientData.Encryption.Enabled" value="true"/>

Decrypted: <add key="Database.PatientData.Encryption.Enabled" value="false"/>.

Changing this command line will result in subsequent records being stored with decrypted data. Records already encrypted may be decrypted using the CONNECT™ Manager Configuration Studio application, as described in the Service Manual [(5), Section 5.7.1, p. 123].

CONNECT™ EPD Structure

CONNECT™ has a number of tables dedicated to storage of various clinical data such as patient details, CPB equipment and disposables, priming solutions, laboratory biochemical data, cardioplegia delivery, procedural details, heart–lung machine, and patient monitoring data. The CONNECT™ tables currently used for EPD transfer into the ANZCPR and description of clinical data contained therein are listed in Table 1. The structure of most of the tables is in wide format, in which each different data variable is stored in a separate column; however, for the PerfusionStreamData table, the different variables from the physiological monitor are stored together in one

Table 1. CONNECT™ tables used for transfer of data into the ANZCPR.

CONNECT Table Name	Data Description	Storage Frequency
CalculationData	Calculated CPB data variables eg; cardiac index, systemic vascular resistance, etc.	Every 20 seconds
CardioplegiaData	Cardioplegia details; type, route, temperature, pressure, volume, etc.	Per dose delivered
CoagulationData	Coagulation test results; activated clotting time, prothrombin time, etc.	Per sample
EventData	CPB events recorded automatically (alarms, timers, etc.) and as entered by the Perfusionist	Per event
GasFlowData	Gas flow data recorded from the electronic blender; gas flows, FiO_2, etc.	Every 20 seconds
LaboratoryData	Blood gas machine data (external to heart–lung machine)	Per sample
MetabolicData	Data for calculated oxygen delivery and carbon dioxide elimination	Every 20 seconds
Patient	Patient details; name, date of birth, gender, etc.	Per patient
PerfusionData	Heart lung machine data; pump flows, pressures, temperatures etc.	Every 20 seconds
PerfusionStreamData	Patient physiologic monitoring data; blood pressures, temperatures, heart rate, etc.	Every 20 seconds
Surgery	Primary key field (SurgeryGuid), and procedural data for each record; date of operation, case record numbers (unique identifiers)	Per procedure
SurgeryAttributeValue	Reference table used to categorize system values	n/a
SurgeryCaseData	Patient and procedural details; height, weight, urgency of procedure, blood type, etc.	Per procedure
SurgeryEquipment	CPB hardware, disposables, cannulae, implants, etc.	Per procedure
SurgeryTeamMember	Operating team member names	Per procedure
SurgeryTeamRole	Reference table used to categorize professions of team members	n/a
SurgeryVolume	Fluid and drug administration or loss values	Per event
TimerData	Timer values for the HLM (CPB, clamp, etc.) and for events defined as having timer values	Per timer

column. To separate individual values, a specific process was developed using Visual Basic for Applications (VBA), the scripting language used within MS Access. The VBA script development to convert the physiological data stream into a data table in wide format was a key component of the EPD integration into ANZCPR. The intraoperative data in the CONNECT™ tables is time-stamped, allowing linkage of the physiological data to the procedural and device data. The unique identifying value for each record within the CONNECT™ database is stored in the "Guid" field in the Surgery table. This value is stored in each table containing procedural data in the "SurgeryGuid" field. In a relational database, this is known as the primary key.

Initiation of the EPD Integration Process

Not all data from each table are transferred to the ANZCPR. All heart–lung machine, cardioplegia, blood gas, and physiological data are transferred; however, all other ANZCPR data-set variables that can be obtained from the CONNECT™ EPD are populated through the use of VBA programing contained within a module in the Transfer database. To activate the transfer of the EPD, the user clicks a button on the current patient record form in the Server database (Figure 2). Clicking the button activates a VBA subroutine that initially checks whether the Transfer database is currently in use, to limit transaction to one record at a time. If the Transfer database is ready to begin a new transaction, the ANZCPR CPB procedure number of the current record is stored in a transaction table in the Tables database. A "transaction in progress" indicator is set and remains in place until data processing and transfer is complete. A configuration table is used to store the location of the Transfer and Tables databases at each participating hospital that can be referenced in the VBA subroutine. The location of the Transfer database is determined, the database is opened, and the EPD integration process is initiated. In order for the VBA subroutine to reference the appropriate libraries (listed in Appendix 1), these should be initialized in MS Access. The VBA subroutine for initiating the EPD process is reported in Appendix 2.

EPD Integration Process

The VBA subroutines and functions for the EPD integration process are contained within a module in the Transfer database. The overall process is controlled from a main subroutine that calls other subroutines and functions to perform specific tasks. The sequence is executed as follows; initially, the ANZCPR CPB procedure number that was stored as part of the EPD initiation process is determined. Using SQL the CPB start and stop times are determined by searching for these data in the CPB event data table (EventData).

The physiological data stream is converted into a data table in wide format. This process activates a separate subroutine for this purpose which

- defines the labels of the field names to search for in the data stream.
- creates a record-set using an SQL query.
- evaluates each stream of data to extract the numerical values for each field name.
- populates a table in the Transfer database with the data.

Because the physiological variables collected at each participating hospital may be different, the subroutine must include each CONNECT™ label used, and these must also be fields in the Transfer database table. Consistency in labeling CONNECT™ physiological variables should be

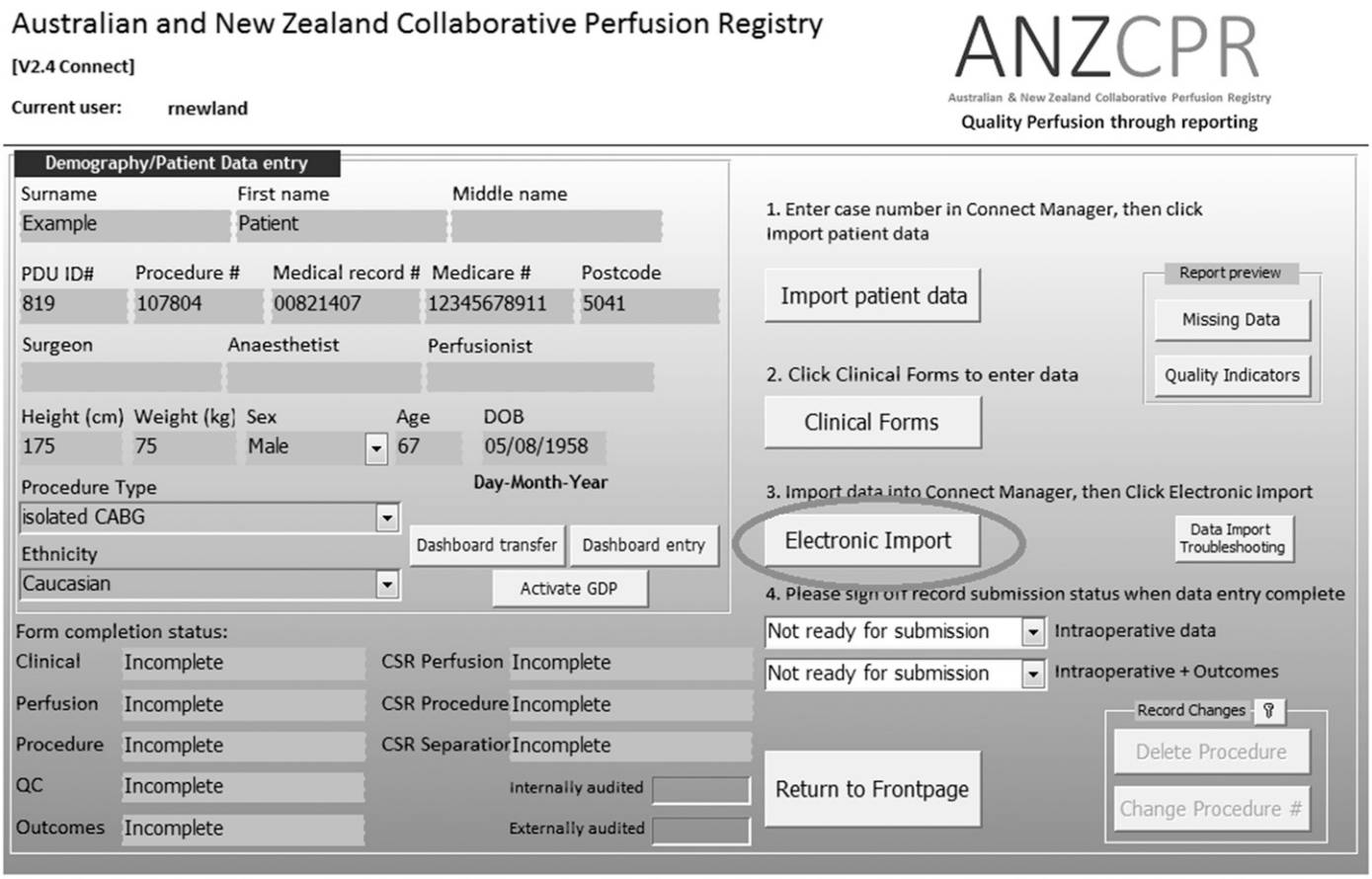

Figure 2. The current patient record form; to transfer data from CONNECT™ to the ANZCPR, the user clicks the electronic import button.

maintained across participating hospitals. This table is empty at the beginning of the EPD integration process. The data from each individual patient record are then used to generate calculated variables and then the complete physiological data-set is appended to a registry table (ConnectPerfusionData) for the storage of multiple patient records. Generating calculated variables creates a single record summary from the multiple data points collected during the bypass period every 20–60 seconds for each procedure. For example, the average cardiac index during CPB is a single data value calculated from the patient's entire record. Storage of the entire physiological data stream in a dedicated table in a wide format is an important step in the integration of EPD with the CONNECT™ system, because it provides storage of the EPD in a format that allows the generation of additional calculated registry variables from the original data if required. The VBA script for the conversion of the physiological data stream is reported in Appendix 3.

EPD variables may be transferred directly from CONNECT™ data fields (e.g., CPB time); however, most are calculated values during the CPB period, such as minimum hemoglobin, average cardiac index, duration that the mean arterial pressure <50 mmHg. SQL queries are then used to create record-sets along with VBA functions to determine the EPD variables according to the ANZCPR EPD data definitions. The main VBA subroutine with examples of how various types of EPD variables are generated is reported in Appendix 4.

To make calculations on the data from electronic perfusion software, and to transfer certain data fields, some conventions common to all participating sites are required; these include "Rewarm" must be included as a comment during the procedure for calculation of temperature parameters during rewarming. "Heparin" and quantity must be entered to define heparin given during CPB. "Partial bypass" can be commented to remove the period of partial bypass from evaluation of cardiac index for quality indicator calculation purposes. The timer labels "Bypass Start" and "Bypass Stop", and "X-Clamp On" and "X-Clamp Off" must be used. In the coagulation table, the sample type "1st ACT post hep" must be used to define the 1st activated clotting time measurement after heparin is administered. Blood gas, hemoglobin, and glucose values transferred from an external blood machine (intermittent sampling) are used for blood gas and electrolyte quality

indicator data. Continuous blood gas data are used for quantification of oxygen delivery if the Spectrum M4 monitor (Spectrum Medical, Gloucester, UK) is used. In this case, arterial flow data is obtained from the M4. Because not all centers use continuous blood gas monitoring with an arterial flow probe, the arterial flow rate from the heart–lung machine is used for calculated cardiac index parameters.

The values returned from each query or function are updated to the appropriate field within the ANZCPR using a specific subroutine reported in Appendix 5. Additional miscellaneous subroutines utilized for the EPD process are reported in Appendix 6. Once the physiological stream data have been processed, the data in the Transfer database table are deleted in preparation for the next patient record to be processed. Finally, the transaction table is reset to allow processing of the next EPD record.

Because CONNECT™ does not allow deletion of event data from the record, errors in manual data entry are marked with "E" in the comment field for each event, which allows exclusion of these events from analysis. A configuration table in the ANZCPR allows for variation in units of hemoglobin measurement, blood gas pressure units, and data collection interval.

ANZCPR Database Elements for EPD Integration

The database tables and field names that are required within the ANZCPR are listed in Table 2. The Tables database has the following four tables that are used for EPD integration:

- The Config table stores the data collection interval and units of measurement for hemoglobin and blood gas values.
- The ItemLocations table stores the location of the Tables and Transfer databases.
- The sysFlags table stores the current status of the EPD integration process (whether a record is currently being processed).
- The ConnectPerfusionData table stores the perfusion stream data in wide format.

The Transfer database has one table (PerfusionStreamData) for temporary storage of the CONNECT™ perfusion data stream to facilitate conversion to wide format. The perfusion data stream is stored permanently in the ConnectPerfusionData table in the Tables database. All other tables accessed by the Transfer database are linked tables stored in the CONNECT™ database or the Tables database.

Table 2. ANZCPR table structure.

Transfer Database

Local Table	Field Name	Data Type	Description
PerfusionStreamData	ID	AutoNumber	Primary key
	CONNECT perfusion stream data variables	Number	CONNECT perfusion stream data variables with one field per variable

Linked Table	Location
Config	Tables database
ConnectPerfusionData	Tables database
ItemLocations	Tables database
sysFlags	Tables database
ANZCPR data-set tables	Tables database
CONNECT tables	CONNECT EPD tables as listed in Table 1

Tables Database

Local Table	Field Name	Data Type	Description
Config	Datatype	Number	Data collection interval in seconds
	Hbunits	Number	Numeric code for either g/dL or g/L
	Gasunits	Number	Numeric code for either kPa or mmHg
ItemLocations	Item	Text	Database name (TablesDb, TranferDb)
	itemLoc	Text	Database file location & filename (e.g. c:\Tables.mdb)
sysFlags	Flagname	Text	Set to 'IsImporting' during EPD transfer
	Flag	Yes/No	Set to 'Yes' during EPD transfer
	Procnum	Text	ANZCPR procedure number during EPD transfer
ConnectPerfusionData	Procnum	Text	ANZCPR procedure number
	Timestamp	Date/Time	Time of data storage in CONNECT
	CONNECT perfusion stream data variables	Number	Storage of perfusion stream data in wide format. One field for each variable
ANZCPR data-set tables	Procnum	Text	ANZCPR procedure number

Currently, 88 of the 310 variables that comprise the ANZCPR data-set are generated from EPD integration. These variables are stored throughout each of the ANZCPR data-set tables. ANZCPR data-set variable definitions are available on the ANZCPR website (www.anzcpr.org).

DISCUSSION

This article describes the process to integrate EPD from the CONNECT™ software into the ANZCPR. Although this method specifically applies to integration of data from the CONNECT™ software into an MS Access database, the principles of EPD integration can be generalized to other perfusion data collection systems and registry datasets. These principles include: creation of registry variables that can be populated with EPD either directly or derived through calculation; procedural record data linkage between the registry and the EPD source database through a unique identifier; development of a process to query the EPD source database and update the registry variables; and additionally, store the EPD in a format that allows access generation of additional registry variables if required. We have reported the main VBA subroutine used with examples of how various types of EPD variables are generated in the ANZCPR. These examples can be generalized to other perfusion registry variables through alteration of the SQL query structure to suit the registry variable definition and the EPD source.

An important consideration in the automated collection of data is limiting the impact of erroneous data. For example, continuous data from online blood gas monitoring devices may be erroneous until calibration of a sample has been performed. Similarly, errors in pressure or temperature measurements may occur. Data processing can be used to limit the influence of erroneous data through the development of specific approaches to each issue, for example, continuous blood gas data can be excluded from analysis until the time that the first calibration sample is received. Data that are clearly outside of normal physiological ranges can also be excluded. Furthermore, in a registry setting, some variation in the accuracy of certain data variables may occur. For example; variation in the accuracy of arterial outlet temperature may be influenced by differences in accuracy of oxygenator temperature probes. Variation in the accuracy of flow rates, used to calculate cardiac index and oxygen delivery may be introduced, either by using fixed arterial pump flow rates for all values irrespective of flow through arterial-venous shunts in the CPB circuit or because of the positioning of ultrasonic flow probes in relation to shunts.

Although the potential for EPD to influence perfusion practice has been demonstrated (1–3), the generalizability of previous reports of EPD integration is limited by superceded software or lack of sufficient detail for reproducibility. The ANZCPR have used EPD to achieve multicenter process improvement as an example of how EPD can be used for generation of CPB quality indicators (QI) (4) to facilitate continuous monitoring of QI parameters and benchmark local performance to other hospitals. The inclusion of EPD in analyses of the impact of CPB on patient outcome is important in improving the understanding of CPB practice on outcome. Perfusion registries play an important role in this process, and the incorporation of EPD into perfusion registries could make a significant contribution toward this objective. By sharing the methodology used to integrate EPD from the CONNECT™ software into the ANZCPR, our intent is to diminish some of the barriers to adoption of EPD integration into other perfusion registries, by providing an example of how EPD integration may be achieved.

REFERENCES

1. Newland RF, Baker RA, Stanley R. Electronic data processing: The pathway to automated quality control of cardiopulmonary bypass. J Extra Corpor Technol. 2006;38:139–43.
2. Baker RA, Newland RF. Continuous quality improvement of perfusion practice: The role of the electronic data collection and statistical control charts. Perfusion. 2008;23:7–16.
3. Stammers AH, Trowbridge CC, Pezzuto J, et al. Perfusion quality improvement and the reduction of clinical variability. J Extra Corpor Technol. 2009;41:48–58.
4. Baker RA, Newland RF, Fenton C, et al. Perfusion downunder collaboration. Developing a benchmarking process in perfusion: A report of the perfusion downunder collaboration. J Extra Corpor Technol. 2012;44:26–33.
5. CONNECT™ Service Instructions.Version 01/2014-SM-45-90-14.02 ENG.

APPENDIX 1

The following libraries should be initialized in MS Access through the Microsoft VBA editing window; VBA

MS Access 14.0 object library
Microsoft DAO 3.6 object library
OLE automation
Microsoft VBA extensibility 5.3
Microsoft scripting runtime
Microsoft activeX data objects 2.8 library
Microsoft activeX data objects recordset 2.8 library

APPENDIX 2

VBA subroutine for activating EPD process for current patient record in ANZCPR.

Note: This process is activated on clicking a button on the database form, stored in the server database.

```
Private Sub cmd_Import_CONNECT_Click()
Dim AccessApp As Object
```

```
Dim curProcnum As String
Dim con As ADODB.Connection
Dim fInTrans As Boolean 'flag for determining if we are currently in a transaction or not
'This checks to see if the transfer database is currently in use
If IsImporting Then
MsgBox "Error, import already in progress, exiting", vbOKOnly + vbCritical, "Error"
Exit Sub
End If
Set AccessApp = CreateObject("Access.Application")
If SysCmd(acSysCmdAccessVer) >= 11 Then
Call AutomateSecurity(AccessApp)
End If
If MsgBox("Starting transfer", vbOKCancel) = 1 Then
SysCmd acSysCmdInitMeter, " Data Transferring ", 10
curProcnum = Me.Procnum
'preset the transaction flag
fInTrans = False
Set con = CurrentProject.Connection
' Start of transaction.
con.BeginTrans
fInTrans = True
con.Execute ("UPDATE sysFlags SET sysFlags.Procnum = '" & curProcnum & "' WHERE sysFlags.FlagName='Importing'")
con.CommitTrans
fInTrans = False
AccessApp.OpenCurrentDatabase DLookup("[itemloc]", "[itemlocations]", "[item] = 'TransferDb'")
SysCmd acSysCmdUpdateMeter, 3
AccessApp.Run "Transfer_CONNECT", UserName
SysCmd acSysCmdUpdateMeter, 6
AccessApp.Quit
Set AccessApp = Nothing
SysCmd acSysCmdUpdateMeter, 10
MsgBox "Finished Transfer"
SysCmd acSysCmdRemoveMeter
Me.Refresh
Else
End If
End Sub
```

APPENDIX 3

VBA script to extract CONNECT™ perfusion data stream

Note: This script is stored in the Transfer database.

```
Private Sub Extract_StreamData(theGUID As String, bpStart As Date, bpEnd As Date, pnum As String)
Dim FieldNameArray As Variant
FieldNameArray = Array("HeartRate", "StAlgorithm1", "StAlgorithm2", "StAlgorithm3", _
"ArtPress_sys", "ArtPress_dia", "MAP", "CVP", "NasoPharynigialTemp") 'Note: this list should include all variable names in the PerfusionStreamData
Dim NumFields As Integer
NumFields = UBound(FieldNameArray) + 1
Dim SQLQuery As String
SQLQuery = "SELECT s.TimeStamp, s.StreamData, d.Bsa " & _
"FROM dbo_PerfusionStreamData s left join dbo_SurgeryCaseData d on d.SurgeryGuid = s.SurgeryGuid " & _
"WHERE s.SurgeryGuid = '" & theGUID & "' and " & _
"s.Timestamp >= #" & Format(bpStart, "yyyy-mm-dd hh:nn:ss") & "# and" & _
"s.Timestamp <= #" & Format(bpEnd, "yyyy-mm-dd hh:nn:ss") & "# " & _
"ORDER BY s.TimeStamp; "
OutputLine ("Processing Perfusion stream data")
Dim stDataRset As DAO.Recordset
Dim stDataTable As Recordset
Set stDataRset = CurrentDb.OpenRecordset(SQLQuery)
stDataRset.MoveFirst
Set stDataTable = CurrentDb.OpenRecordset("PerfusionStreamData", dbOpenTable)
Do While Not stDataRset.EOF ' read each record
stDataTable.AddNew
stDataTable("ProcNum").Value = pnum
stDataTable("Timestamp").Value = stDataRset!TimeStamp
Dim fcnt As Integer
For fcnt = 0 To (NumFields - 1) ' parse out each of the fields and put in table
Dim fname As String
fname = FieldNameArray(fcnt)
If (GetFieldValue(fname, stDataRset!StreamData) <> "") Then
Dim fval As Variant
fval = CDbl(GetFieldValue(fname, stDataRset!StreamData))
If (fname = "ArtTemp") Then ' ensure values are within a valid range
If (fval < 5) Or (fval >= 100) Then
fval = Null
End If
End If
If (fname = "Hb") Then ' ensure values are within a valid range
If (fval < 1) Or (fval >= 100) Then
fval = Null
End If
End If
If fname = "pCO2Art_37" Then ' as above
If (fval < 5) Or (fval >= 120) Then
fval = Null
End If
End If
If Not IsNull(fval) Then ' if a value was found then
```

```
stDataTable(fname).Value = fval ‘store it in the perfusion data table field
End If
End If
Next fcnt
stDataTable.Update
stDataRset.MoveNext
Loop
End Sub
‘This function is to extract the field names from the perfusion data stream
‘The string containing the fields and their values is formatted such that an ASCII 30 character ‘(record separator) surrounds each record, within which an ASCII 31 character (field separator) ‘separates the field name from its value
Public Function GetFieldValue(fldName As String, fldStr As String) As String
Dim namePos As Integer
Dim valPos As Integer
Dim valLen As Integer
Dim nextSep As Integer
Dim fldVal As String
namePos = InStr(1, fldStr, fldName + Chr(31))
If IsNull(namePos) Or (namePos <= 0) Then
GetFieldValue = ""
Else
valPos = namePos + Len(fldName) + 1
nextSep = InStr(valPos, fldStr, Chr(30))
valLen = nextSep - valPos
fldVal = Mid(fldStr, valPos, valLen)
GetFieldValue = fldVal
End If
End Function
```

APPENDIX 4

VBA subroutine for the EPD integration process in ANZCPR.

Note: This script is stored in the Transfer database.

```
Public Sub Transfer_CONNECT(logusername As String)
Dim logProcnum As String
'lookup current CPB procedure number:
logProcnum = DLookup("Procnum", "sysFlags")
'check that the database is not currently processing data:
If IsImporting Then
MsgBox "Error, import already in progress, exiting", vbOKOnly + vbCritical, "Error"
Exit Sub
Else
CurrentDb.Execute ("UPDATE sysFlags SET sysFlags.Flag = True WHERE sysFlags.Flagname='Importing'")
End If
'retrieve key values that we will need in queries:
Dim procGuid As String
Dim patientGuid As String
Dim surgDate As Date
'run subroutine to retrive the values:
Call GetSurgDetails(logProcnum, procGuid, patientGuid, surgDate)
If procGuid = "" Then
MsgBox "Procnum " & logProcnum & " cannot be found in CONNECT database - no update done."
Exit Sub
End If
OutputLine ("GUID = " & procGuid)
' define CPB start and stop times
Dim bySQL As String
Dim byRSet As Recordset
Dim bypassStart As Date ' get time of first bypass start
Dim nullBypassStart As Boolean
extraData.bypassStartTime = Null
nullBypassStart = True
bySQL = "select min(EventTime) as StartTime From dbo_EventData " & _
"where SurgeryGuid = '" & procGuid & "' and " & _
" ((CommentText is null) or (CommentText <> 'E')) and " & _
" ((SourceLabel = 'Bypass' and EventLabel = 'Start') or (SourceLabel = 'Bypass Start'))"
Set byRSet = CurrentDb.OpenRecordset(bySQL)
If Not byRSet.EOF Then
If Not IsNull(byRSet![StartTime]) Then
bypassStart = byRSet![StartTime]
extraData.bypassStartTime = byRSet![StartTime]
nullBypassStart = False
End If
End If
byRSet.Close
Dim bypassEnd As Date ' get time that last bypass ends
Dim nullBypassEnd As Boolean
nullBypassEnd = True
bySQL = "select max(EventTime) as EndTime From dbo_EventData " & _
"where SurgeryGuid = '" & procGuid & "' and " & _
" ((CommentText is null) or (CommentText <> 'E')) and " & _
" ((SourceLabel = 'Bypass' and EventLabel = 'Stop') or (SourceLabel = 'Bypass End') or (SourceLabel = 'Bypass Stop'))"
Set byRSet = CurrentDb.OpenRecordset(bySQL)
If Not byRSet.EOF Then
If Not IsNull(byRSet![endTime]) Then
bypassEnd = byRSet!endTime
nullBypassEnd = False
End If
End If
byRSet.Close
' clear the perfusion stream data table from last import
```

```
CurrentDb.Execute "delete * from [PerfusionStreamData];"
' now populate table
Call Extract_StreamData(procGuid, bypassStart, bypassEnd, logProcnum)
'This section contains examples of how to generate calculated EPD variables
'Here is an example of retrograde autologous prime volume
'The comment RAP is entered as a volume comment and the volume amount is retrieved
Dim RapVol As Integer
RapVol = 0
bySQL = "select Value as RapValue From dbo_EventData " & _
"where SurgeryGuid = '" & procGuid & "' and " & _
" ((CommentText is null) or (CommentText <> 'E')) and " & _
" (SourceLabel = 'RAP') and SourceType = 'Volume -' "
Set byRSet = CurrentDb.OpenRecordset(bySQL)
If Not byRSet.EOF Then
RapVol = Nz(byRSet!RapValue, 0)
End If
byRSet.Close
'Here are examples of how to extract minimum and maximum blood gas and electrolyte data, 'incorporating different units of hemoglobin and blood gas pressure. Also a binary
'quality indicator is set for having a blood glucose <4 or > 10
Dim minHb As Single
Dim maxHb As Single
Dim minCO2 As Single
Dim maxCO2 As Single
Dim minGlucose As Single
Dim maxGlucose As Single
Dim Gluc As Integer
OutputLine ("Extracting Laboratory max/mins")
labSQL = "select Min(Hb_Ext) as MinHb, Max(Hb_Ext) as MaxHb, " & _
" Min(pCO2Art_37_ext) as MinCO2, Max(pCO2Art_37_ext) as MaxCO2, " & _
" Min(Glucose_ext) as MinGlu, Max(Glucose_ext) as MaxGlu " & _
" from dbo_LaboratoryData " & _
" where SurgeryGuid = '" & procGuid & "' and " & _
" TimeStamp >= #" & Format(bypassStart, "yyyy-mm-dd hh:mm:ss") & "# and " & _
" TimeStamp <= #" & Format(bypassEnd, "yyyy-mm-dd hh:mm:ss") & "#"
Set labRSet = CurrentDb.OpenRecordset(labSQL)
If Not labRSet.EOF Then
If Nz(DLookup("Hbunits", "Config")) = 1 Then
minHb = labRSet!minHb * 10
maxHb = labRSet!maxHb * 10
ElseIf Nz(DLookup("Hbunits", "Config")) = 2 Then
minHb = labRSet!minHb
maxHb = labRSet!maxHb
Else
MsgBox ("Bad config value for HbUnits")
Exit Sub
End If
If Nz(DLookup("Gasunits", "Config")) = 1 Then
minCO2 = labRSet!minCO2
maxCO2 = labRSet!maxCO2
ElseIf Nz(DLookup("Hbunits", "Config")) = 2 Then
minCO2 = labRSet!minCO2 * 7.5
maxCO2 = labRSet!maxCO2 * 7.5
Else
MsgBox ("Bad config value for HbUnits")
Exit Sub
End If
minGlucose = labRSet!minGlu
maxGlucose = labRSet!maxGlu
If minGlucose < 4 Or maxGlucose > 10 Then
Gluc = 1
Else
Gluc = 2
End If
End If
labRSet.Close
'Here are examples of how to extract min and max values from the perfusion stream data table
Dim perfSQL As String
Dim perfRSet As Recordset
OutputLine ("Extracting Perfusion min/max/avgs")
perfSQL = "SELECT Min([NasoPharynigialTemp]) AS [MinOfNaso temp], " & _
"Max([NasoPharynigialTemp]) AS [MaxOfNaso temp], " & _
"Max([ArtTemp]) AS [MaxOfArt temp], " & _
"Min([ArtTemp]) As [MinOfArt temp], " & _
"Min(Nz([MAP])) AS [MinOfArt P], " & _
"Avg([MAP]) AS [AvgOfArt P] " & _
"FROM PerfusionStreamData " & _
"WHERE ProcNum = '" & logProcnum & "'"
Dim Nasomin As Single
Dim Nasomax As Single
Dim MAPavg As Double
Dim Artmax As Single
Dim Artmin As Single
Set perfRSet = CurrentDb.OpenRecordset(perfSQL)
If Not perfRSet.EOF Then
Nasomin = perfRSet![MinOfNaso temp]
Nasomax = perfRSet![MaxOfNaso temp]
MAPavg = perfRSet![AvgOfArt P]
Artmax = perfRSet![MaxOfArt temp]
Artmin = perfRSet![MinOfArt temp]
End If
```

```
perfRSet.Close
'Here is an example of how to extract an average value from the perfusion data table
perfSQL = "SELECT Avg([ArtFlow]) AS [AvgOfArt Flow] " & _
"FROM dbo_PerfusionData " & _
"WHERE SurgeryGuid = '" & procGuid & "' and " & _
" TimeStamp >= #" & Format(bypassStart, "yyyy-mm-dd hh:mm:ss") & "# and " & _
" TimeStamp <= #" & Format(bypassEnd, "yyyy-mm-dd hh:mm:ss") & "#"
Dim Flowavg As Double
Set perfRSet = CurrentDb.OpenRecordset(perfSQL)
If Not perfRSet.EOF Then
Flowavg = perfRSet![AvgOfArt Flow]
End If
perfRSet.Close
'To update the values returned from the queries into a registry table, use this syntax, 'according to the registry table and field name; 'eg updSQL = "UPDATE [Registry table name] SET [Registry field name] = " & [vba variable name] & "
Dim updSQL As String
updSQL = "UPDATE ANZCPR_Perfusion SET Hbmin = " & minHb & ", " & _
"Hbmax = " & maxHb & ", " & _
"Nasomin = " & Nasomin & ", " & _
"Nasomax = " & Nasomax & ", " & _
"Artmax = " & Artmax & ", " & _
"Artmin = " & Artmin & ", " & _
"MAPavg = " & MAPavg & ", " & _
"Flowavg = " & Flowavg & ", " & _
"Glucmin = " & minGlucose & ", " & _
"Glucmax = " & maxGlucose & _
" WHERE Procnum = '" & logProcnum & "'"
doUpdate (updSQL)
'Here is an example of how to calculate cumulative time variables and then taking into account the data collection interval
'This is the cummulative time that the MAP was < 40 mmHg; Dim artP40 As Single
countSQL = "SELECT COUNT(TimeStamp) As TheCount from PerfusionStreamData where (MAP >= 30) and (MAP < 40) and procnum = '" & logProcnum & "'"
artP40 = getCount(countSQL) ' Note: getCount is a function decribed later on
'This is the cummulative time that the arterial outlet temperature was >37 degrees; Dim ATemp37 As Single
countSQL = "SELECT COUNT(Timestamp) As TheCount from PerfusionStreamData where (ArtTemp > 37) " & " and procnum = '" & logProcnum & "'"
ATemp37 = getCount(countSQL)
'Determining the data collection interval from the configuration table; Dim BDQCfactor As String
If Nz(DLookup("Datatype", "Config")) = 2 Then
BDQCfactor = "0.5"
Else
OutputLine ("BD - QC Update 20 sec")
BDQCfactor = "0.33333333"
End If
Dim countSQL As String
'Updating the data taking into account the data collection interval; updSQL = "UPDATE ANZCPR_PerfQC set artP40 = " & artP40 & "*" & BDQCfactor & ", " & _
"[ATemp>37] = " & ATemp37 & "*" & BDQCfactor & _
" WHERE Procnum = '" & logProcnum & "'"
doUpdate (updSQL)
'append physiological stream data to multiple record table
If IsNull(DLookup("[ProcNum]", "ConnectPerfusionData", "[ProcNum] = '" + logProcnum + "'")) Then
CurrentDb.Execute "Append Stream data"
End If
'now clear out the temporary stream data table and reset the transaction table
CurrentDb.Execute "delete * from [PerfusionStreamData];"
Dim MySQL As String
CurrentDb.Execute ("UPDATE sysFlags SET sys Flags.Procnum = Null WHERE sysFlags.FlagName= 'Importing'")
CurrentDb.Execute ("UPDATE sysFlags SET sys-Flags.Flag = false WHERE sysFlags.FlagName= 'Importing'")
MsgBox "Transfer complete"
Exit Sub
End Sub
```

APPENDIX 5

VBA subroutine to update values returned in the SQL queries to the database fields

Note: This script is stored in the Transfer database.

```
'This subroutine is used to update values returned in the SQL queries to the database fields
Public Sub doUpdate(SQLtoUse As String)
Dim updTable As String
If Len(SQLtoUse) > 50 Then
Dim wordArray() As String
wordArray() = Split(SQLtoUse)
Select Case wordArray(0)
Case "UPDATE"
updTable = wordArray(0) + " " + wordArray(1) + " ..."
Case "INSERT"
updTable = wordArray(0) + " " + wordArray(1) + " " + wordArray(2) + " ..."
End Select
```

```
End If
With CurrentDb
.Execute SQLtoUse
If .RecordsAffected <= 0 Then
Call OutputLine("Update was not successful: " & SQLtoUse, True, updTable)
End If
End With
End Sub
```

APPENDIX 6

```
Miscellaneous specific VBA subroutines
Note: These scripts are stored in the Transfer database.
'this function finds the Connect unique record identifier for the corresponding ANZCPR unique procedure identifier, together with the date of surgery
Public Sub GetSurgDetails(Procnum As String, ByRef procGuid As String, ByRef patientGuid As String, ByRef surgDate As Date)
Dim guidSQL As String
Dim guidRset As Recordset
guidSQL = "Select * from dbo_Surgery where CaseNumberDec = '" & Procnum & "'"
Set guidRset = CurrentDb.OpenRecordset(guidSQL)
If guidRset.RecordCount > 0 Then
procGuid = Mid(StringFromGUID(guidRset!Guid), 8, 36) ' get the Connect GUID for this procedure and chop off extraneous characters added by routine
patientGuid = Mid(StringFromGUID(guidRset!patientGuid), 8, 36) ' The CONNECT identifier for the patient
surgDate = guidRset!SurgeryDate
Else
Call OutputLine("Error finding procnum " & Procnum & " in CONNECT database", True)
procGuid = ""
End If
guidRset.Close
End Sub
'This function determines if the transfer database is currently in use
Public Function IsImporting() As Boolean
If DLookup("[Flag]", "[sysFlags]", "[FlagName]='Importing'") = True Then
IsImporting = True
Else
IsImporting = False
End If
End Function
'This function is used to return the number of records for a certain criteria (e.g., count number of times pressure <40)
Private Function getCount(SQLtoUse As String) As Integer
Dim cntRset As Recordset
Dim retcount As Integer
Set cntRset = CurrentDb.OpenRecordset(SQLtoUse)
If cntRset.RecordCount > 0 Then
retcount = cntRset!theCount
Else
retcount = 0
End If
cntRset.Close
Set cntRset = Nothing
getCount = retcount
End Function
```