Iphasoft Pty Ltd

2 Azalea Place
LOFTUS NSW 2232
(612) 9576 6381 0409 357 551
sales@alphasoft1.com
ABN 17 002 661 421

Flexible Information Display System notes

9 Feb 2011

The FIDS design is based on the following requirements :-

1. Meet time performance criterion of ~10mSec per screen format on incremental update.
2. Must be robust.
3. Must be international esp Chinese
4. Must be developed for multiple Windows versions and be portable to Linux without too much cost.
5. Must be easy to install and manage, including graphic screen layout.
6. Must support the existing Video generator boxes and screen formats.
7. Must be flexible and inexpensive to maintain.
8. Should be applicable to other application areas of electronic signage.
9. Allow for automation of testing.
10. View and edit database (DB) overall from an administrator tool on any server.
11. View an approximation of any display on a server.
12. View feed activity on any server.
13. View system summary information, esp connected clients / operators.

How these requirements have been met :-

1) High speed data base and data transport. Database announces changes to registered handlers. Formatter 'pre-compiles' screen descriptions to binary, in-memory versions for high speed display. Supports incremental display – only changes part of display required.
2) Multiple PCs can be data base mirrors ready to become the server if required. The display formatting is automatically distributed among the available cores/PCs. The code is as small as possible. The functions can be distributed over multiple .exe files reducing the damage done by a single failure and maximising deployment flexibility.

   The multi-thread model is simple, well defined, and restricted to the IP and pipe interface area. Application programmers do no have to provide additional locking.

   The data base code size is small and “readable”, so should be reliable. The data base has received extensive testing.

   The only libraries are the Delphi VCL which is well tested and the Indy winsock library which seems to be stable enough.

   Data transport and tree data base functions are encapsulated into the MessageHub and MirrorDB units/classes and their ancestor classes, for easy use by applications.

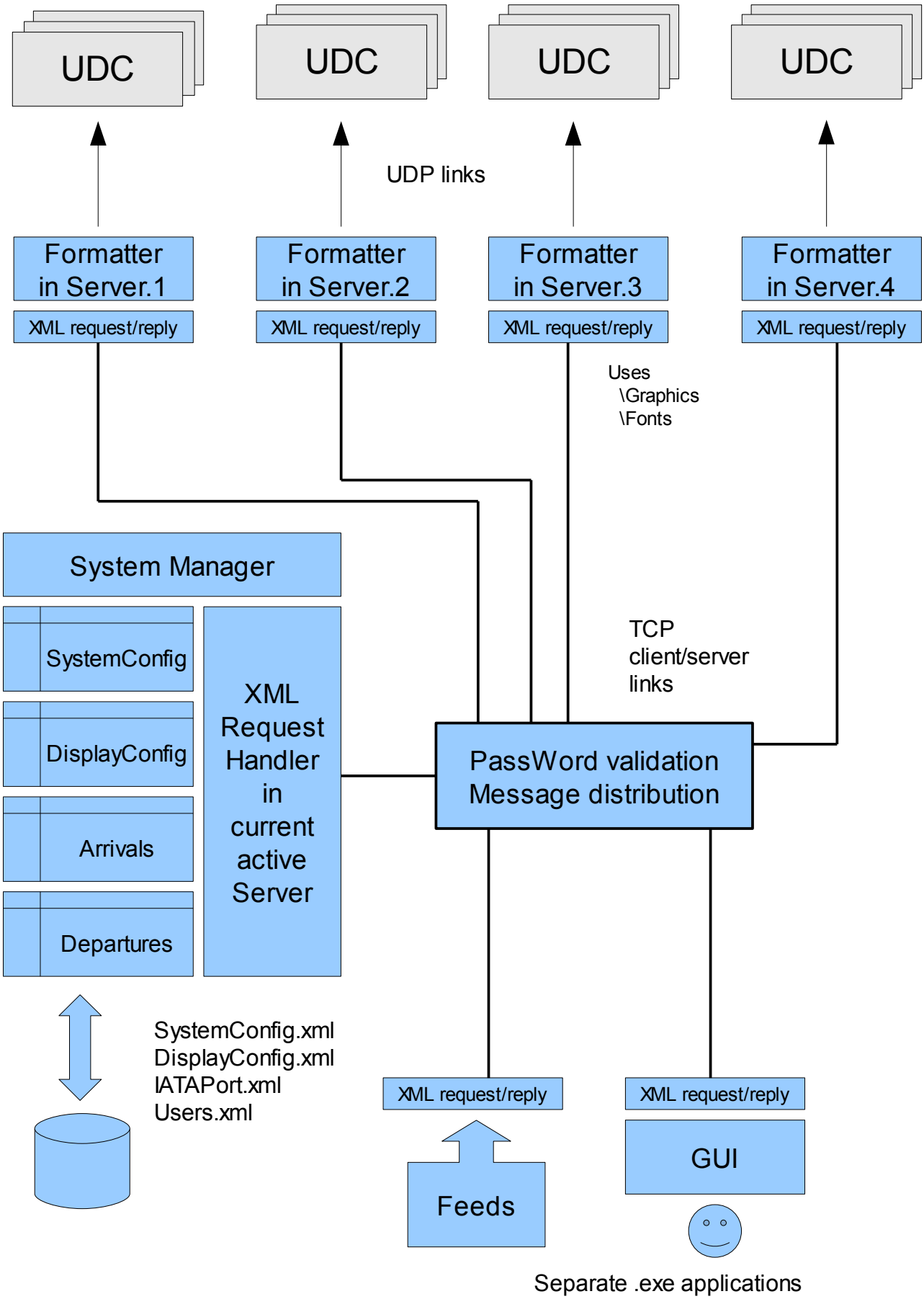
3) User interface ultimately uses Window's unicode functions for text display. Internal transport uses XML UTF-16 to suit Windows unicode API. XML files are stored and transferred in UTF-8.
4) Core components are written in Delphi 2009/Object Pascal that can be compiled on Linux (Free Pascal), with minimal modification. Delphi 2011 is expected to cross compile to native linux.
5) The application doesn't need to use the registry. Install is simply copy and run. All configuration is done through the configuration data base so can be changed remotely or by some user interface elements. The application sorts itself out as much as possible.
6) The display configuration is handled by its own data base which can support the standard UDC options.

7) The system is well instrumented :-
   - The communication format is XML which is human readable, simple to parse, and largely self describing.
   - The saved to disk version of a data base is also XML. The application programs then refer to the field by name at run time so adding new fields is straightforward.
   - There is extensive error logging,
   - and a network traffic viewer to simplify debugging.
   - There is a viewer for display formats and formatter performance and traffic statistics.
   - There is a generic data tree display and editing administrator tool.
   - A readable log of the UDC drive can be logged.

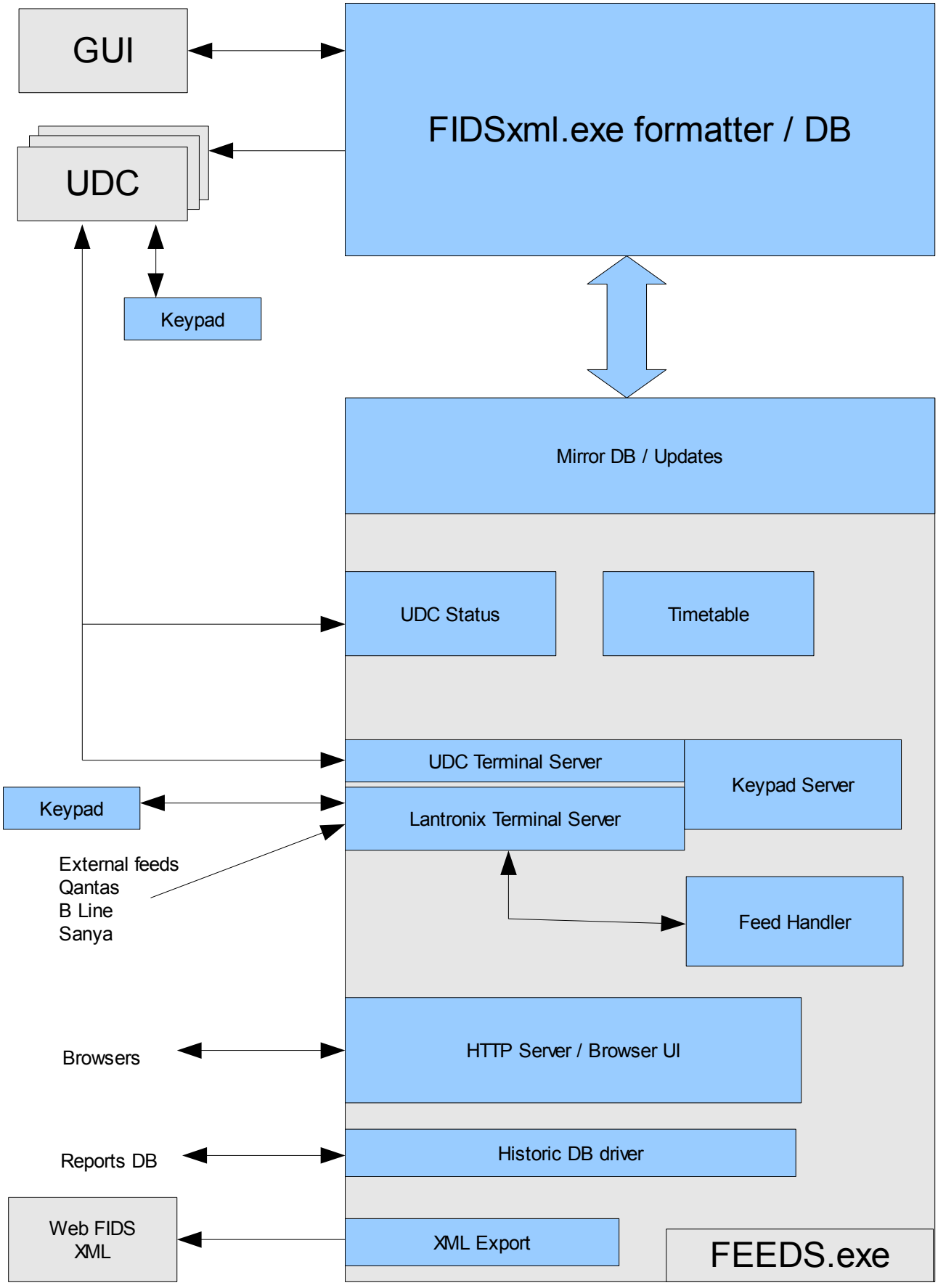
   - The format specification is humane (xml – reasonably high level of abstraction and readable-ish).

   The system is designed to have a well defined layered implementation :-
   - Transport layer
   - Messaging layer
   - Database layer
   - Data type verification layer
   - Data entry layer
8) The system structure and formatter capabilities lend themselves to application areas other than airports.
9) The whole system is text stream in and out including the UDC commands. File in / file out and automated testing is possible for much of the system. This could be performed after small changes to quickly verify that most of the system still works.

FIDSxml FORMATTER AND DB SERVER BLOCK DIAGRAM



FEEDS BLOCK DIAGRAM



The Feeds program

Airport specific functions have been separated from the main database/formatter program, into a separate .exe, to improve software modularity. The Feeds.exe includes automatic flight removal, time-table flight generation, external SQL database connection, UDC polling and startup detection, keypad operation, and any external feeds.

Feeds.exe only runs on the Host PC and can provide a watchdog function on FIDSxml.exe. (todo)

Automatic flight removal can be disabled in SystemConfig.xml and has various parameters in SystemConfig.xml. Removed flights can be written to a log file or exported to an external SQL database to support report generation (todo).

The Flight generation timetable can be disabled in SystemConfig.xml and has various parameters in SystemConfig.xm. The main parameter is look-ahead (typically 16 hours) which controls how far ahead new flights are created. The timetable runs once a minute and creates flights with scheduled times 16 hours from now. A Timetable 'Reset' control is provided to force the timetable to scan the timetable rules for the whole 16 hour period and create all required flights. This would recreate any accidentally deleted flights or pick-up any timetable additions that would generate a current flight. The timetable rules are read from Timetable.xml. New flights are created using the 'Template' part of the flight rules and are subject to the Validation unit acceptance before being added to the database. The timetable will not alter an existing flight in the database.

UDC Polling manages the SystemSettings|DeviceStatus|UDC|NotResponding part of the database which can be reported by any UI components.(todo) UDCs are deemed to be not responding if they don't accept a TCP connection on port 80. (The UDP poll is not currently supported on UDC-4s.)

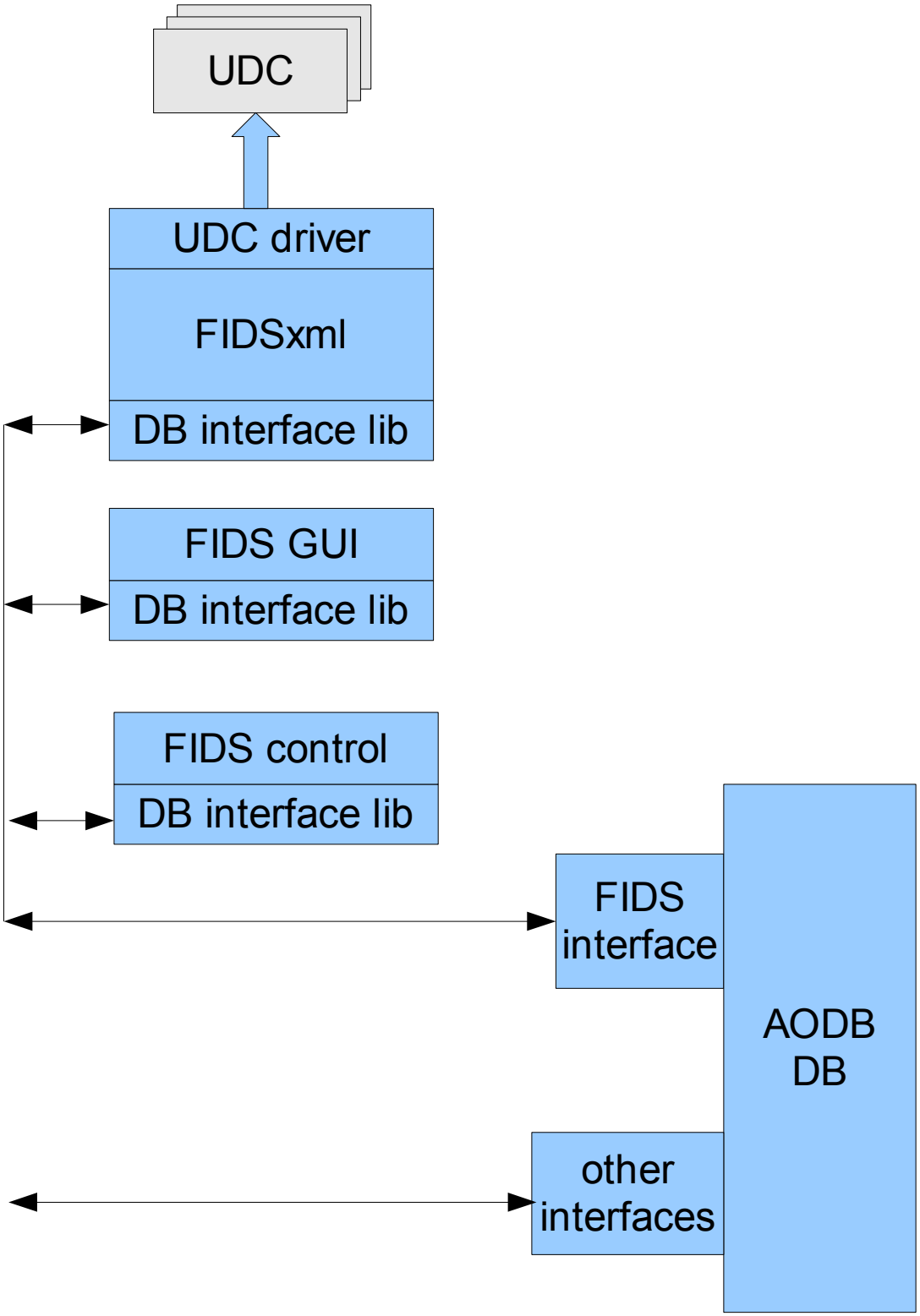
UDC starup detection provides a TCP listener port (9201) to detect any UDCs being turned on after Host initialization. This sends a <UDCStartup> message to the formatter to reinitialize that IP.(todo)

The keypad operation requires the collection of serial input from UDC and Lantastic 'terminal servers' and updating the flight database or the CheckIns style database. UDC terminal servers are driven by sending <UDCSerial> messages to the formatter's UDC driver. (The UDCs automatically poll their keypads every 1/2 second giving a very sluggish user response). Keypad status information is kept in SystemSettings|DeviceStatus|Keypad|NotResponding part of the database (todo).

All external feed input is subject to new flight validation. Keypad status information is kept in SystemSettings|DeviceStatus|Feed_name_|NotResponding part of the database (todo).

FIDSxml as a component of AODB

Because FIDSxml is designed as an open system it can be integrated into a complete AODB by replacing a specific feed, such as Qantas, with a generic connection to the AODB as its source of flight information. The FIDS specific user interface is provided by 'FIDS GUI' and any FIDS specific requirements are implemented in 'FIDS control'.



Some Implementation Conventions

1. The user interface elements do all input checking.
   This reduces the complexity of the data base without significant cost.
2. Display elements are responsible for sorting.
3. Use standard tag names so the data base name space is predictable.
4. Tag names should be legal Pascal identifier names.
5. DB content is stared as it is to be displayed.
6. The root tag descendants are loaded from and saved to individual xml files.
7. '|' and '#' characters must not be in DB content (escape them).
8. Root tags are Arrivals, Departures, SystemConfig, DisplayConfig
9. The only xml attributes used are KeyTag=”true” (forces unique descendant names) and AutoInc="55" (creates unique descendant names by appending an incrementing postfix to the child node names eg QFA421-646)
10. Flight names are made unique. (see 9 above)
11. DB security is handled by the user interface elements using name/password information from DB \SystemConfig\Security\ information
12. Server configuration uses DB \SystemConfig\Servers\Server.1 etc.
13. Display configuration is based on DB \DisplayConfig\Frames\... and \DisplayConfig\UDC\..... \IATAPort\SYD\... etc.
14. The formatter accepts a XML specification for each frame type (see example later), and generates the required output as a series of rectangular fields that be sent to pretty much any graphic device. Simple @functions are used to make attribute and text adjustments to suit airport use. Doing specials will usually just require coding a new function.

Installation Specifics

Currently the install base diectory is FIDSxml. This contains the following :-
FIDSxml.exe is the server executable that is compatible with the Win32 platforms ie Windows 2000, XP and Vista. It is resonsible for database manipulations and UDC screen updates. Nearly all communication to the UDCs is over UDP packets (mostly broadcast to frame addresses) on IP. All other net comms (ie to GUI, feed etc) is via TCP on IP. This other traffic is all XML coded text – see example below.

*.xml are text files containing XML format data loaded and used by FIDSxml.exe

FIDSxml.log is a text file added to by FIDSxml.exe to log various FIDS errors and events

FLIGHT DATA BASE FIELDS

	FIDSxml Tag	Qantas	Sanya	B-Line		
Clear all flights	N/A	manual	<RESET> <dateTime>			
Flight header	tree	Message delimited	<FLIGHT>			
			<reasonCode>			
	<Arrivals>	AF	<type>			
	<Departures>	DF	<type>			
Non-flight data		NF				
FlightKey	<NZ123-765654>	FLTN	(composite)			
		Qflight				
subflight/code shr	<Flights> (tree)		<flightNumber>			
			<primaryFlt>			
		PFLT				
	(Corresponding primary)	Qflight				
	<Terminal>		<terminal>			
	<CarrierType>	FLTT	<carrierType>			
		D/I				
	<NonPublic>	PDFL	<public>			
		Y/N				
	<Ports>	ROUT	<orign> <destination> -<via>			
			--<via1..5>			
	IATA List	AAA/AAA/..				
	<CheckIns>		<checkins>			
	3,4,5		-<counters>			
			-<openD..T>			
			-<closeD..T>			
	<ST>	SKTM	<SD> <ST>			
		DateTime				

	<ET>	COTM	<ED> <ET>			
		DateTime				
	<AT>	TATM	<AD> <AT>			
		DateTime				
	<Aircraft>	ATYP	<aircraftType>			
		S(3)				
	<Rego>	AREG	<registration>			
		S(7)				
	<Gates>	GAT1	<gates>			
	2,3	S(3)	-<gates>			
			-<openD..T>			
			-<closeD..T>			
		GAT2				
		S(3)				
	<Bays/>	PBAY	<bay>			
	<Comment>	PUDC	<publicComment1..2>			
	<StaffComment>	STDC	<staffComment1..2>			
			<crawlingLine1..2>			
		STDN				
	<AStatus>	FLST	<status>			
		A(1)				
	<DStatus>	FLST	<status>			
		A(1)				
	See <DStatus>	BDST				
	<PassAdult/>	PAXC				
	<PassInfant/>	N(3)/N(3)				
	<LateralNumber>		<lateralNumber>			
NFI			<blockD..T>			
			<baggageType>			
			<exitNo>			
(ARRIVALS)						
	<CorrespondingFlight>	NXOP	<relatedFlt>			
	FlightKey	Qflight				
	<Belts>	CAR1	<beltNumbers>			
	list					
	<Belt>		<belt>			

			-<beltNumber>			
	<FirstDrop>		-<firstBagD..T>			
	<LastDrop>		-<lastBagD..T>			
		CAR2				
	<BeltPriority>	CARP				
	n	N				
	<DisPassAdult/>	PDIS	<totalPax>			
	<DisPassInfant/>	N/N				
	<TransitPassAdult>	PTRN				
	<TransitPassInfant>					
	<TranshipPassAdult>	PTRS				
	<TranshipPassInfant>					
	<MeetAssist>	MAAS				
	<UnaccompaniedMin>	UMIN				
	<YoungPeople>	YPER	<youngPAX>			
	<WheelChair>	WCHR	<wheelchairPAX>			
(DEPARTURES)						
	<Fuel>	FUEL				
	<Burn>	BURN				
	<AirCraftConfig>	ACFG				
		n/n/n...				
Pre boarding	<PreBrd>	PBTM				
		DateTime				
boarding	<Board>	BDTM				
		DateTime				
Final cal	<FinalCall>	FCTM				
		DateTime				
locked	<Closed>	LCTM				
		DateTime				
	<TransitPassAdult>	PTRN				
	<TransitPassInfant>					

	<JoiningPassAdult>	PJNR				
	<JoiningPassInfant>					
		BNOP				
Normal Closing time		BNCL				
Peak open time		BPOP				
Peak closing time		BPCL				
Late closing time		BLCL				
Default sort lateral		BDFL				
Default on-carriage sort lat.		BDOL				
Sort criteria for terminating bags		BP1T				
		(and 2..8)				
Sort criteria for connecting bags		BP1C				
		(and 2..8)				
		AUDN				
		Record no.				

Qflight - Qantas A fully qualified flight number. Format is: S(2..3)/N(1..4)S/YYMMDD/N
FlightKey – QF123-195296 unique flight designator

Sample Request/Response Traffic

```
<DataRequest>
        <ReqRead>  | </ReqRead>
        <Path>  |Users|Alphasoft| </Path>
        <ReqID> 2 </ReqID>
</DataRequest>
<DataReply>
        <Result> OK
                <ReqRead> | </ReqRead>
                <Path> |Users|Alphasoft| </Path>
                <ReqID> 2 </ReqID>
        </Result>
        <Users>
                <Alphasoft>
                        <PassWord/>
                        <Access/>
                        <LoggedIn/>
     </Alphasoft>
   </Users>
</DataReply>


<EditRequest>
        <ReqEdit>  |Hello again|Hello| </ReqEdit>
        <Path>  |Users|Alphasoft|PassWord| </Path>
        <ReqID> 2 </ReqID>
</EditRequest>
<EditReply>
        <Result> OK
                <ReqEdit> |Hello again|Hello| </ReqEdit>
                <Path> |Users|Alphasoft|PassWord| </Path>
                <ReqID> 2 </ReqID>
        </Result>
</EditReply>


<EditRequest>
        <ReqEdit>  |5|| </ReqEdit>
        <Path>  |Departures|ST1234-20090114|Bay| </Path>
        <ReqID> 2 </ReqID>
</EditRequest>
<EditReply>
        <Result> ERROR 105
                <ReqEdit> |5|| </ReqEdit>
                <Path> |Departures|ST1234-20090114|Bay| </Path>
                <ReqID> 2 </ReqID>
        </Result>
</EditReply>


<LoginRequest>
        <ReqID> GUI </ReqID>
        <UserName> ST </UserName>
        <PassWord> Password </PassWord>
</LoginRequest>
<LoginReply> Success   <Access> 9,* </Access> </LoginReply>
```

Sample XML Files and descriptions

see DefinitiveXML.xml for most up to date FIDS tree conventions.

SystemConfig.xml

This is the key job configuration file and must be installed with all FIDS related executables. It contains the license information that allows the release versions of the executables to run. It contains the server IP addresses that allows the system parts to find each other and communicate over the network. It contains the names of job specific resources like gate names.

```
<?xml version="1.0" encoding="UTF-8"?>
<SystemConfig>
	<JobName> Digital Images </JobName>
	<Servers KeyTag="true">
		<TCP_Port> 1666 </TCP_Port>
		<Server.1>
			<IP> 192.168.0.163 </IP>
		</Server.1>
		<Server.2>
			<IP> 192.168.0.164 </IP>
		</Server.2>
		<Server.3>
			<IP> 192.168.0.2 </IP>
		</Server.3>
		<Server.4>
			<IP> 192.168.0.4 </IP>
		</Server.4>
	</Servers>
	<Terminals> T1,T2,T3 </Terminals>
	<Bays>
		1,2,3,4,5,6,7,8,9
	</Bays>
	<Gates>
		1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
	</Gates>
	<Counters>
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32
	</Counters>
	<Belts> 1,2,3,4,5,6,7,8,9 </Belts>
</SystemConfig>


<?xml version="1.0" encoding="UTF-8"?>
<SystemSettings>
	<DisplayFonts>
		<FontPath> D:\Windows\Fonts\ </FontPath>
		<Font>
			Bitstream Cyberbit
			<Range> $4E00-$9FFF </Range>
		</Font>
	</DisplayFonts>
	<Logo>
		<QF>
			<Range> 100-299
				<Use> AR </Use>
			</Range>
			<Alt> QFA </Alt>
		</QF>
	</Logo>
```

```
        <UDPinterval> 10 </UDPinterval>
        <Comments/>
</SystemSettings>
```

Logo section is to allow flight number ranges to use a different logo, or some alternate 3 letter airline code to be used as the logo selector.

```
<?xml version="1.0" encoding="UTF-8"?>
<Users>
        <Feed>
                <PassWord> _DI_system_ </PassWord>
                <Access> 9,* </Access>
                <LoggedIn/>
        </Feed>
        <Server>
                <PassWord> _DI_system_ </PassWord>
                <Access> 9,* </Access>
                <LoggedIn/>
        </Server>
        <ST>
                <PassWord> Password </PassWord>
                <Access> 9,* </Access>
                <LoggedIn> 081212  1107 </LoggedIn>
        </ST>
</Users>
```

The users DB holds user names and passwords for login authentication.

```
<?xml version="1.0" encoding="UTF-8"?>
<Departures KeyTag="true" AutoInc="20">
        <QF002-20081114>
                <Ports> MEL,TYO,PEK,SHA </Ports>
                <NonPublic> 1 </NonPublic>
                <STD> 20081114 013000 </STD>
                <ETD> 20081114 083000 </ETD>
                <Terminal/>
                <Gate> G1 </Gate>
                <CarrierType> D </CarrierType>
                <DStatus> Boarding </DStatus>
                <Aircraft> 717 </Aircraft>
                <Comment>
                        □□□□□□  and some more messages
                </Comment>
                <Flights>
                        <QF002>
                                <Counters/>
                                <Ports> MEL,TYO,PEK,SHA </Ports>
                        </QF002>
                        <GT002>
                                <Ports> MEL,TYO,SYD </Ports>
                                <Counters/>
                        </GT002>
                </Flights>
                <Bay/>
        </QF002-20081114>
```

Departures hold all departure flight data.
The supported attributes are KeyTag="true" which enforce uniqueness of sub-tags and fast look-up using a hash table, and AutoInc="20" which appends an auto-incrementing suffix to any un-numbered tags added.

```
<?xml version="1.0" encoding="UTF-8"?>
<DisplayConfig>
        <Formats KeyTag="true">
                <Departures>
                        Public
                        <Rect> 0,0,1359,767, </Rect>
                        <ColourBG> 0,0,0, </ColourBG>
                        <ColourFG> 255,255,255, </ColourFG>
                        <Font> swis721 blkcn bt </Font>
                        <Attributes> -LnBox, -Bold </Attributes>
                        <Procedure> SortDB( :Departures:; STD: ) </Procedure>
                        <Frame>
                                <Header>
                                        <ColourBG> 0,85,85, </ColourBG>
                                        <Line>
                                                <Height> 60 </Height>
                                                ……..
```

All screen formats are specified in DisplayConfig.xml.

The Formatting Process

THE FORMATTING PROCESS - a 4 step procedure

1 - precompile the format spec for each required frame
- in :- displayconfig.xml
- out :- draw instruction tree

2 - expand abstract draw instructions to specific vales using current database values.
- in :- departures.xml etc
- out :- current drawn list of fields

3 - check against currently displayed draw field list and mark changed fields
- in :- previous drawn tree
- out :- changed fields marked

4 - send changes to output devices
- in :- output device control
- out :- current drawn list of fields

Supported Screen Specifications tag names :-

'Formats'	Holds the named format specs.
{name}	eg <Gate> names the format and may have the 'PagedSequence' flag to change multi page mode to support one page for each gate, check-in, baggage, etc. The 'Public' flag controlls the @DStatus() function
'Procedure'	Any supported procedure. Will evaluate either at the beginning of the format or for each page depending on placement.
Filter	Limits the flights that are displayed to those passing the filter tests. The format is [DB path] [relation operator] [value]. eg <Filter> *:Junk: *= Pass </Filter> relators supported are '=', '<>', and '*=' (not defined or equal)
'Frame'	Holds the spec for a rectangular region of a format. They hold bands. 'PagedSequence', 'Public' flags control paging logic and status filtering
	Band tags hold vertically arranged elements. They hold lines.
'Header', 'Footer'	Specifies a part at the top / bottom of each page.
'Body'	holds middle content typically spreading through multiple pages. It typically takes a DB parameter to be repeated for each contained item.
'Line', 'LineIf', 'LineElse'	Specifies a vertical part of a band. It holds fields. Conditional line displays if its argument string is not empty. Deprecated – use If/Else
'Field'	width or 0; text. Specifies a horizontal part of a line. A field is the atom of the display specification. It embodies text or graphics or list of these; to be displayed in a (small) rectangular space on the screen according to the accumulated attributes and other specifications.
FieldIf	width, text, condition. Only drawn if 'condition' text resolves to not blank. Deprecated – use If/Else
FieldElse'	width, text. As 'Field' but only drawn if previous 'FieldIf' failed
'FloatingField'	x, y, width, height, text. Like a field but is positioned anywhere within the line after the last 'Field'. Must not be first field on a line. Must not have width 0.
'If', 'Else'	can be used in Frame, Band, Line or Field context to make that section conditional.

The various draw settings inherit down through the above hierachy of screen space specifiers.

'Rect'	x0, y0, x1, y1 pixel position of a rectangle, applicable to formats and frames.
'ColourFG', 'ColourBG', 'ColourAlt'	R,G,B colour specification. FG foreground, BG background, Alt alternating band background if BGColourAlt is set
'Font'	font name eg 'Arial' or 'Times'. Arial with +Bold attribute will use the UDC internal font 0.
'FontSize'	font size in pixels.
'Height', 'Width'	pixels. Set line height or field width.
'GapH', 'GapV'	pixels. Set the offset of text from the top left corner of the field rectangle.
'Attributes'	+/- attribute_name or attribute list

```
<Groups>                slide show control ( also in DisplayConfig.xml )
    <Dep-Pic>           any name for group
        <Format> Departures,1,5 </Format>          | format name | page number | seconds
        <Format> Departures,2,5 </Format>
        <Image> \Graphics\GIFs\Misc\ttopcl.gif, 5 </Image>          | file name | seconds |
    </Dep-Pic>
</Groups>
```

will change the Departures page 1 displays to slide show mode, page 1 for 5 seconds, page 2 for 5 seconds and the graphic for 5 seconds

Supported Attributes :-

'Blink', 'Scroll', 'Wipe', 'Direction', 'Bold', 'Italic', 'Underline', 'BGColourAlt',
'Center', 'Right', 'VCenter', 'LnBox', 'LnTop', 'LnLeft', 'LnBottom', 'LnRight', 'File', 'LocaliseFont', UpperCase,
Gap, 'PrFirst', 'PrLast'

LocaliseFont uses <SystemSettings>...<Font> to select alternative fonts for non ASCII (foreign) text.
Gap Inserts a space into the flight name between leading alpha characters and numeric characters.
PrFirst', 'PrLast control the parts fitting algorithm giving priority to the first (for arrivals) or last (for departures) or both.

Supported String Functions :-

Field text can contain at functions that are evaluated each time the screen is updated to handle getting information from the data base or performing translation logic.

@Time	windows time/date specifier string eg @Time(h:nn)'
@Page	current page number
@DB'	returns DB content by name eg @DB(:SystemConfig:JobName:)
@ListDB	returns a list from the DB eg
@DBTag	
@ListDBTag	returns a list of DB tag names suitable for handling code shares as a Vscroll display eg @ListDBTag(:MyGate:[1]:Flights:*:)
@FromDBList	returns a single item from a comma separated list.
@AirLogo	converts the field to a file (graphic) field based on the flight code (or list) in it parameter.
@IATAList	converts a list of IATA codes from param1 and param2 to fitted text with language specifier param 3.
@IsForeignIATAList	based on above. Returns empty if no foreign text required. (Use in LineIf)
@List	converts its params into a list suitable for Vscrolling.
@File	use param1 as a file name (.GIF).
@Astatus	adjust 'on blocks' to 'Arrived' – subject to 'Public' flag
@Dstatus	adjust param1 as a departure status, ie blink 'final call' – subject to 'Public' flag
@SubStr	extract characters from param 1 eg
@CheckInTop, @CheckInBottom	handles logic for top and bottom bands of checkin displays involving the <Counters><1><Logo> RE </Logo> <Service> Re-Checkin </Service> <Class> sections of the data base.
@DBCount	returns the number of subnodes of a database item. Typically allows the screen layout to vary based on the number of flights. Eg <If> @DBCount(*; 1) ie if 1 flight.

Supported Procedures :-

SortDB	base DB address; field to sort by. Eg <Procedure> SortDB(:Departures:; STD:) </Procedure> sorts the departures flights into scheduled time of departure order.
SearchDB	temp DB name; base DB name; Field to match; DB list ; abd index of match value Eg. SearchDB(MyGate; :Departures:; Gate; :SystemConfig:Gates:; @Page()) Makes a temp DB called MyGate. It is a subset of the departures flights, selected if their 'Gate' branch (field) matches the gate name for gate number 'page'. Typically used in summary type displays such as Gate displays.

UDC Internal Fonts :-

```
PreLoad( 'ttf.ttf', 'ttf' );    // 0 UDC preloaded fonts  (Arial Bold)
PreLoad( 'tires.ttf', '' );
PreLoad( 'lucida.ttf', '' );
PreLoad( 'lucidab.ttf', '' );
PreLoad( 'swissm.ttf', '' );
PreLoad( 'Swis721 BlkCn BT.ttf', 'swissbc.ttf' );
PreLoad( 'Bitstream Cyberbit.ttf', 'cyberbit.ttf' );  // 6
```

The default font is Arial bold – font 0.

Checkin 'Logic'

(top of screen - airline logo area)

if <DisplayConfig>...<attributes> +CheckInLogoOverride, then
show <Counters>...<Logo>

else if number of flights = 0 then
show <Counters>...<Logo>

else show @AirLogo(@ListDBTag(*:Flights:[1]:)) ie logo of first flight

---

(middle)

flight or list of flights - the usual stuff

(bottom of screen - service logo area)

if <Counters>..<Service> show <Counters>..<Service> gif
else if <Counters>..<Class> then show airlinecode + class_code .gif
else some default

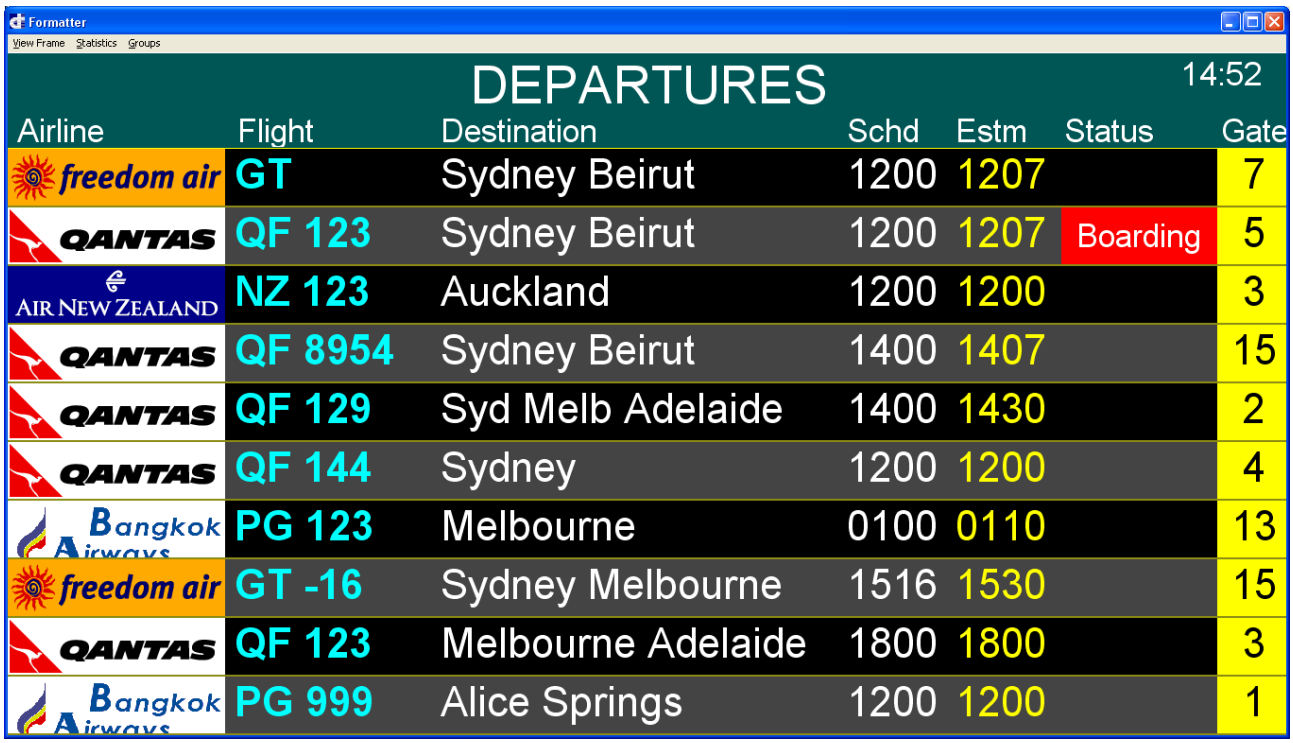
FORMATTER EXAMPLE

The following format specification produces the illustrated 'Departures' frame from test data :-

```
<?xml version="1.0" encoding="UTF-8"?>
<DisplayConfig>
	<Formats KeyTag="true">
		<Departures>
			Public
			<Rect> 0,0,1359,767, </Rect>
			<ColourBG> 0,0,0, </ColourBG>
			<ColourFG> 255,255,255, </ColourFG>
			<Font> swis721 blkcn bt </Font>
			<Attributes> -LnBox, -Bold </Attributes>
			<Procedure> SortDB( :Departures:; STD: ) </Procedure>
			<Frame>
				<Header>
					<ColourBG> 0,85,85, </ColourBG>
					<Line>
						<Height> 60 </Height>
						<Attributes> +Center </Attributes>
						<Field> 150, </Field>
						<Field>
							0,DEPARTURES
							<FontSize> 55 </FontSize>
						</Field>
						<Field>
							150,@Time(hh:mm)
							<FontSize> 34 </FontSize>
						</Field>
					</Line>
					<Line>
						<Attributes> -Center </Attributes>
						<Height> 40 </Height>
						<FontSize> 34 </FontSize>
						<Field>
							240,Airline
							<GapH> 10 </GapH>
						</Field>
						<Field> 215,Flight </Field>
						<Field> 430,Destination </Field>
						<Field> 115,Schd </Field>
						<Field> 115,Estm </Field>
						<Field> 165,Status </Field>
						<Field>
							80,Gate
							<Attributes> +Center </Attributes>
						</Field>
					</Line>
					<Line>
						<Height> 1 </Height>
						<ColourBG> 255,255,0 </ColourBG>
						<Field/>
					</Line>
				</Header>
				<Body>
					:Departures:
					<FontSize> 44 </FontSize>
					<ColourBG> 0,0,0, </ColourBG>
					<ColourAlt> 68,68,68, </ColourAlt>
					<Height> 60 </Height>
```

```
                                        <Line>
                                                <Attributes> +BGColourAlt </Attributes>
                                                <Field> 230,@AirLogo(@ListDBTag(*:Flights:*:)) </Field>
                                                <Field>
                                                        225,@ListDBTag(*:Flights:*:)
                                                        <Font> SwissM </Font>
                                                        <ColourFG> ,0,255,255, </ColourFG>
                                                        <GapH> 10 </GapH>
                                                        <Attributes> +Gap, +Bold </Attributes>
                                                </Field>
                                                <Field>
                                                        430,@IATAList( *:Flights:*:Ports:; *:Ports: )
                                                        <Attributes> +LocaliseFont,+PrLast,-PrFirst
</Attributes>
                                                </Field>
                                                <Field>
                                                        230,@DStatus(*:DStatus:)
                                                        <If> @DStatus(*:DStatus:) </If>
                                                        <FontSize> 34 </FontSize>
                                                        <Attributes> +Center </Attributes>
                                                        <GapV> 10 </GapV>
                                                        <ColourBG> 0,255,0 </ColourBG>
                                                </Field>
                                                <Field> 115,@SubStr(@DB(*:STD:); 10; 4)
                                                        <Else/>
                                                </Field>
                                                <Field>
                                                        115,@SubStr(@DB(*:ETD:); 10; 4)
                                                        <Else/>
                                                        <ColourFG> 255,255,0, </ColourFG>
                                                </Field>
                                                <Field>
                                                        165,@DStatus(*:DStatus:)
                                                        <FontSize> 34 </FontSize>
                                                        <Attributes> +Center </Attributes>
                                                        <GapV> 10 </GapV>
                                                </Field>
                                                <Field>
                                                        80,@DB(*:Gates:)
                                                        <Attributes> -BGColourAlt, +Center </Attributes>
                                                        <ColourFG> 0,0,0, </ColourFG>
                                                        <ColourBG> 255,255,0, </ColourBG>
                                                </Field>
                                        </Line>
                                        <Line>
                                                <If> @DB(*:Comment:) </If>
                                                <FontSize> 34 </FontSize>
                                                <Attributes> +Scroll, +BGColourAlt, +LocaliseFont </Attributes>
                                                <Field> 0,@DB(*:Comment:) </Field>
                                        </Line>
                                        <Line>
                                                <Height> 2 </Height>
                                                <ColourBG> 140,140,20 </ColourBG>
                                                <Field/>
                                        </Line>
                                </Body>
                        </Frame>
                        <Devices>
                                <Page>
                                        1
                                        <UDC>
```

```
                                        <IPs>
                                                192.168.00.20
                                                <Location> Departure Hall E17 </Location>
                                        </IPs>
                                </UDC>
                        </Page>
                        <Page>
                                2
                                <UDC>
                                        <IPs>
                                                192.168.00.254
                                                <Location> Departure Hall E17 </Location>
                                        </IPs>
                                </UDC>
                        </Page>
                </Devices>
        </Departures>  </Formats>
</DisplayConfig>
```


Formatter
View Frame Statistics Groups
DEPARTURES
14:52
Airline Flight Destination Schd Estm Status Gate
freedom air GT Sydney Beirut 1200 1207 7
QANTAS QF 123 Sydney Beirut 1200 1207 Boarding 5
AIR NEW ZEALAND NZ 123 Auckland 1200 1200 3
QANTAS QF 8954 Sydney Beirut 1400 1407 15
QANTAS QF 129 Syd Melb Adelaide 1400 1430 2
QANTAS QF 144 Sydney 1200 1200 4
Bangkok Airways PG 123 Melbourne 0100 0110 13
freedom air GT -16 Sydney Melbourne 1516 1530 15
QANTAS QF 123 Melbourne Adelaide 1800 1800 3
Bangkok Airways PG 999 Alice Springs 1200 1200 1

BUILD ENVIRONMENT, DEPENDENCIES

The current development system is Delphi 2007. It uses the standard VCL library, Indi 10 TCP components. Source is held for all these. The only binary dependence is Windows. The test platforms are Windows 2003 Server, Windows XP Pro and Windows 7.

User interface components could be implemented in anything that can connect to a TCP server and exchange UTF-8.

Testing is facilitated by a random flight feed generator and the ability to display to a Windows window. This means that most development can be done on a PC without needing a bank of UDCs or an airline feed. FIDSxml incorporates a database viewer / editor and a traffic monitor.

UDC INTERFACE

Each defined frame and page is defined a unique frame number and is updated using the UDP-broadcast- frame address mode (FIDS mode) supported by the UDC.

The UDC imposes a numbered storage scheme. Typically the design allocates numbers to screen objects in the form <page>.<frame>.<line>.<field_no>
<page> are assigned from 1 for each defined time multiplexed sub frame.

PROGRAM INTERFACE

The SDK provides some generic interface classes. For data transport see cMessageHub that supports the sending of xml strings and the asynchronous reception of xml string responses. cXMLParser facilitates the creation of request xml strings and parsing of xml to data trees. cMirrorDB allows complete mirroring of the database (as a data tree) in the local process memory.

These classes support either a simple client/server model or an async local memory resident mirrored database with delta notification. An example of the latter is provided in the SDK, see GUI.dproj. There are many routines available to make reading and searching the data tree fast. To make global changes, there are many formatting routines available. Typical User program operation is to :-

1. Log onto the database server – initiating a TCP connection, and getting user access permissions.
2. Scan some part of the database that is relevant into a local data tree.
3. Display the data to the user.
4. Handle delta notifications by updating user displays.
5. Make the required changes using the New, Edit and Delete edit request messages.
6. Log off – ending the TCP connection.
7. The server will broadcast the changes to the other data mirrors, and persist the data as xml files on the server machine.

(FIDS initial design ~Feb,Mar 08, PC prototype to Jul 08, UDC driver to Sep 08, Screen layout enhancements to Mar 09, database and data transport development to Dec 09, FEEDS as separate project/application to Feb 10, extensions to screen description language to Mar 10 GT/Alphasoft)