Preliminary Traffic Cop Specification

This paper provides an early look at the Desktop Integration Library's Traffic Cop (TCOP) and explores the early implications for developers who may plan to use this application.

The traffic cop will be an application (or device driver, if required by Copland) that will act as a serial port arbitrator and application manager for DIL applications. This is a simple application that has two very rudimentary modes : (1) set-up, and (2) management. These modes will probably be exclusive.

TCOP Set-up Mode

This mode will consist of a simple dialog that displays a list of applications that can be launched by the TCOP. The user can add or delete applications from this list. In adding, the user chooses a menu item or an ADD button and will be presented with a standard file dialog that will allow selection of an application to add. This application's full name (including path) will be added to the list. The user can also choose to select a nickname for this application. The nickname will generally be the name of the application without the path, and perhaps other information. For example, the full name might be "Eggman:Applications:Newton Connection Kit for Macintosh" and the nickname might be "Newton Connection" or "NCK." To delete an application, a user might select the application to delete and choose a menu item or a DELETE button.

While in the set-up mode, TCOP will also allow the user to select "Preferences…" from the menu. This will bring up a dialog that will allow selection of automatic listening on launch and specification of communications port options.

TCOP Management Mode

In this mode, TCOP will be monitoring or arbitrating serial ports, or managing applications. These functions are best described by presenting a typical simple traffic cop scenario:

1. At set-up, a user selects applications that can be launched by the TCOP and sets some basic preferences. This data will be written out to resources, Windows registry or a preferences file.

At TCOP launch, the TCOP will determine the ports available, but will only grab ports to listen if the user preferences indicate that the TCOP should listen for incoming calls or connections at all times.

At this point, there are two possible usage models.

2a. In the first model, the user can launch one or more TCOP-aware applications. These applications will simply make DIL calls to a TCOP driver by specifying "TCOP:<connectiontype>" in the CDPipeInit call, where

<connectiontype> is a valid serial communications transport. For example, an application might call CDPipeInit with the connection type “TCOP:MNP.” This tells the DIL libraries that the traffic cop is to be used and MNP serial is the low-level transport. The CDPipeInit call will register these applications as ACTIVE with the TCOP. When the DILs disconnect, this will deregister this application. If an application that is registered crashes, the TCOP will first attempt to talk to the application through the callbacks provided by the application. If these fail, the TCOP will assume the application has died, and will launch the application (as in scenario 2b).

2b. In the second model, the TCOP will wait for a request for an application and will launch that application upon request.

To accomplish either model, the Newton application dials the phone. When TCOP answers, the Newton application will then send a TCOP kTCOPRequestForApplication message to the TCOP. All TCOP messages will consist of a four-byte prefix(‘TCOP’), a four byte message identifier (in this case, ‘TRFA’), a four-byte data length field (if there is data accompanying this type of message), and data (in this case, the application full name or nickname). For example, to launch or talk to NCK by the nickname “NCK”, the Newton application should send “TCOPTRFA\0\0\0\3NCK”

In the first model, the TCOP will set the appropriate CDIL_State (kCDIL_ConnectPending) in the space of the appropriate application (if synchronous), or will call back to the application (if asynchronous) to let this application know that there has been a connection. In the second model, the TCOP will launch the application, wait for the application to register and go to a listening state, and then will perform the above steps. If the application launch fails, the TCOP will send back the message kTCOPApplicationFailed, along with a four byte error code, if any (“TCOPTAFD\0\0\0\4\0\0\0\1”). If launch is successful, the TCOP will return kTCOPSuccess. This will also let the Newton application know that it is talking to a TCOP and not directly to the application. A TCOP imitator (an application that will accept TCOP messages, but is not a TCOP) should return kTCOPNotHere to let the application know that TCOP messages will not cause a problem with this application. A non-response or an error response should be taken by the Newton application as a sign that there is no TCOP or TCOP-aware application running.

3. Once the application is connected, it will read and write using the CDPipeRead and CDPipeWrite methods just as it would using another transport. A Newton application will also just read and write data as usual. The TCOP will examine the first four bytes in the incoming (from the Newton to the desktop) stream ONLY for any TCOP messages, and then pass the data on to its destination. The desktop application will not be required to send the TCOP messages, The desktop application will be sending implicit

messages to the TCOP though DIL calls.

4. When the Newton application is ready to disconnect from the current desktop application, it can either send a kTCOPDisconnect message or simply issue another kTCOPRequestForApplication message. This will allow the TCOP to inform the desktop application (through DIL state changes) that it should clean up. The TCOP does not have to wait for the first application to disconnect to launch the next application and start talking to it , since the first application does not control the physical port. Once the first application has issued the last normal CDIL call (CDDisconnect), the TCOP is free to kill this application, if it was launched by the TCOP.

Newton developers need to create a globally available endpoint that others can share,with a global usage semaphore to indicate whether the endpoint is in use currently. A TCOP-aware Newton application should NOT always create a new endpoint at start-up or close the endpoint at disconnect, as is commonly done today. Instead, on start-up that application should check to see if the global endpoint is in use. If it is not, it should instantiate the endpoint as is done today. On shutdown, the Newton application should ask the user whether or not to close the endpoint. If the user chooses to stay connected, the closing application will reset the semaphore, and close without shutting down the communications. The user will then probably launch another Newton application. This new application will need to check the global semaphore and use the endpoint that is in the global slot instead of instantiating a new endpoint. There may be an application that will perform these tasks for packages in the future.

Messages:

The following are the four-byte identifiers proposed for the TCOP:

kTCOPPrefix	'TCOP'
kTCOPRequestForApplication	'TRFA'
kTCOPApplicationFailed	'TAFD'
kTCOPSuccess	'TCOK'
kTCOPNotHere	'TCNH'
kTCOPDisconnect	'TDSC'