Manual: FDO91 Manual
Chapter 42: Asynchronous Data Protocol defines the atoms that manage the asynchronous transfer of large blocks of data between the host and client PCs.
Last updated: October 1997

CHAPTER 42
Asynchronous Data Protocol (ADP)

The Asynchronous Data Protocol (ADP) (protocol ID 66) consists of atoms that manage the transfer of large blocks of data (typically images) between the host and client PCs. The ADP protocol is currently used for e-mail images but can be used for other applications that require downloading or uploading any large block of data. This interactive protocol enables the host to divide a large block of data into smaller chunks that are asynchronously transmitted host to client or client to host. Essentially, the host regulates the flow of a large block of data it receives or transmits to a rate that it can properly process by capitalizing on pauses between the chunk transmissions. An interactive example of an asynchronous upload session is shown in the **atom$adp_start** examples and an asynchronous download session is shown in the **atom$adp_end** examples.

Asynchronous Data Protocol Atoms

The Asynchronous Data Protocol (ADP) functions and their associated atoms follow:

Function	**Atoms**
Establishing or defining an asynchronous data transfer session	atom$adp_abort_tkn atom$adp_chunk_count atom$adp_chunk_size atom$adp_eid atom$adp_eid_string

	atom$adp_route_tkn atom$adp_tranaction_type
Starting or ending an ADP code stream	atom$adp_end atom$adp_start
Sending a data chunk	atom$adp_data atom$adp_echo atom$adp_eid atom$adp_eid_string
Restricting data flow or requesting more data	atom$adp_chunk_count atom$adp_chunk_size atom$adp_echo atom$adp_more_chunks atom$adp_no_more chunks
Terminating a data transfer session	atom$adp_abort atom$adp_no_more chunks

The ADP atoms are described in alphabetical order in the rest of this chapter.

atom$adp_abort
10 ($0A)

Description

atom$adp_abort terminates the data transfer session before it is complete.

Syntax

```
atom$adp_abort
```

Return Value

None.

Example

The following client-to-host example terminates the upload data transfer before it is complete:

```
atom$adp_start
atom$adp_eid <1>
➪ atom$adp_abort
atom$adp_end
```

atom$adp_abort_tkn
5 ($05)

Description

atom$adp_abort_tkn defines a host server routing token the client should use to abort the current data transfer session.

Syntax

```
atom$adp_abort_tkn <word>
```

`<word>`	Specifies a hex value (2 bytes) that defines a host server routing token the client should use to abort this session. The value is a code of 2 ASCII characters. For example, the value 4Ex 57x defines a routing token of NW.

Return Value

None.

Example

The following host-to-client example defines a host server routing token the client should use to abort this session:

```
atom$adp_start
atom$adp_transaction_type <2>
atom$adp_eid_string <M0039813>
atom$adp_route_tkn <4Ex 57x>
➪ atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```


atom$adp_chunk_count
7 ($07)

Description

atom$adp_chunk_count specifies the number of consecutive data chunks that can be sent from the client without waiting for a host signal to continue. Note that **atom$adp_more_chunks** must be sent by the host before more chunk transmission can continue.

Syntax

```
atom$adp_chunk_count <dword>
```

`<dword>`	Specifies the number of consecutive data chunks to send without a signal from the host to continue. Values are 1 or greater.

Return Value

None.

Example

The following host-to-client example specifies the number of consecutive data chunks that can be sent without waiting for a host signal to continue. In this example, the client must wait for a signal (**atom$adp_more_chunks**) from the host after each chunk is sent.

```
atom$adp_start
atom$adp_transaction_type <1>
atom$adp_eid <1>
atom$adp_chunk_size <1900>
➪ atom$adp_chunk_count <1>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```

atom$adp_chunk_size
6 ($06)

Description

atom$adp_chunk_size specifies the chunk size of each data segment to be transmitted. Typically, the last chunk in the chain of the data object is smaller than the specified chunk size.

Syntax

```
atom$adp_chunk_size <dword>
```

`<dword>`	Specifies the chunk size in bytes for the data segments to be transmitted. Values are 1 or greater.

Return Value

None.

Example

The following host-to-client example sets the chunk size of the data segments to 1900 bytes:

```
atom$adp_start
atom$adp_transaction_type <1>
atom$adp_eid <1>
➪ atom$adp_chunk_size <1900>
atom$adp_chunk_count <1>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```


atom$adp_data
8 ($08)

Description

atom$adp_data specifies a chunk of the object data to send to the host or client in an asynchronous transfer session.

Syntax

```
atom$adp_data <raw_data>
```

`<raw_data>`	Specifies the raw chunk of data to send. Note that the chunk size cannot exceed the value specified by **atom$adp_chunk_size**.

Return Value

None.

Examples

The following host-to-client example sends a chunk of data to the client in a download session:

```
atom$adp_start
atom$adp_transaction_type <2>
atom$adp_eid_string <M0039813>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
.
.
atom$adp_start
atom$adp_eid_string <M0039813>
atom$adp_echo <4Ex 57x>
➪ atom$adp_data <34x 54x F9x 02x 34x 54x F9x 34x 54x F9x … >
atom$adp_end
```

The following host-to-client example establishes an upload session with the client. Note that the host sends the following code stream first before the client sends data:

```
atom$adp_start
atom$adp_transaction_type <1>
atom$adp_eid <1>
```

```
atom$adp_chunk_size <1900>
atom$adp_chunk_count <1>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```

The following client-to-host example sends a chunk of data to the host in an upload session:

```
atom$adp_start
atom$adp_eid <1>
➪ atom$adp_data <00x 01x C3x 1Ax 35x 00x 02x … >
atom$adp_end
```


atom$adp_echo
12 ($0C)

Description

atom$adp_echo defines a host server routing token (echo token) the client should use regarding the acknowledgment of data chunks received. Also, the client sends this atom to the host when it is ready to receive more download data.

Syntax

```
atom$adp_echo <raw_token>
```

`<raw_token>`	Specifies a hex value (2 bytes) that defines an echo token the client should use regarding the acknowledgment of data chunks. The value is a code of 2 ASCII characters. For example, the value 4Ex 57x defines a routing token of NW.

Return Value

None.

Examples

The following host-to-client example establishes a download session, sends a data chunk, and defines an echo token for the client to use after it receives the data chunk:

```
atom$adp_start
atom$adp_transaction_type <2>
atom$adp_eid_string <M0039813>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
.
.
atom$adp_start
atom$adp_eid_string <M0039813>
➪ atom$adp_echo <4Ex 57x>
atom$adp_data <34x 54x F9x 02x 34x 54x F9x 34x 54x F9x … >
atom$adp_end
```

The following client-to-host example acknowledges receipt of a chunk of the data object M0039813 using an echo token of 4Ex 57x. This action in a download protocol is called echoing the EID (embedded ID of the object).

```
atom$adp_start
atom$adp_eid_string <M0039813>
➪ atom$adp_echo <4Ex 57x>
atom$adp_end
```


atom$adp_eid
3 ($03)

Description

atom$adp_eid defines the embedded ID (EID) of the data object (such as an image) for transfer to the host. This EID is established by the client for both the host and client to track the interactive code stream sequences throughout an object transfer.

Syntax

```
atom$adp_eid <dword>
```

`<dword>`	Specifies the EID of the data object. Values are 1 or greater.

Return Value

None.

Example

The following host-to-client example defines an EID of 1 to the data object for an upload session:

```
atom$adp_start
atom$adp_transaction_type <1>
atom$adp_eid <1>
atom$adp_chunk_size <1900>
atom$adp_chunk_count <1>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```

atom$adp_eid_string
11 ($0B)

Description

atom$adp_eid_string defines the embedded ID (EID) of the data object (such as an image) for transfer to the client. This EID is established by the host for both the host and client to track the interactive transmission sequences throughout an object transfer session.

Syntax

```
atom$adp_eid_string <string>
```

`<string>`	Specifies the EID of the data object. Values are a sequence of alphanumeric characters (12 maximum).

Return Value

None.

Example

The following host-to-client example defines an EID (M0039813) for a data object of the download session:

```
  atom$adp_start
  atom$adp_transaction_type <2>
➪ atom$adp_eid_string <M0039813>
  atom$adp_route_tkn <4Ex 57x>
  atom$adp_abort_tkn <4Ex 57x>
  atom$adp_end
```


atom$adp_end
2 ($02)

Description

atom$adp_end marks the end of an ADP code stream that may start, continue, or stop an asynchronous data transfer session. Note that each ADP code stream must be started with **atom$adp_start**.

Syntax

```
atom$adp_end
```

Return Value

None.

Examples

The following examples show a set of ADP code streams that illustrate the interactive asynchronous protocol for a download session from the host. Each code stream must be terminated with **atom$adp_end**.

The following host-to-client example ends an ADP code stream that establishes an asynchronous data transfer session with the client:

```
atom$adp_start
atom$adp_transaction_type <2>
atom$adp_eid_string <M0039813>
atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
➪ atom$adp_end
```

The following host-to-client example ends an ADP stream that sends a chunk of data to the client:

```
atom$adp_start
atom$adp_eid_string <M0039813>
atom$adp_echo <4Ex 57x>
atom$adp_data <34x 54x F9x 02x 34x 54x F9x 34x 54x F9x … >
➪ atom$adp_end
```

The following client-to-host example ends an ADP stream that acknowledges receipt of the last data chunk and is ready for more data:

```
  atom$adp_start
  atom$adp_eid_string <M0039813>
  atom$adp_echo <4Ex 57x>
➪ atom$adp_end
```

The following host-to-client example ends an ADP stream that tells the client the data transfer session is complete:

```
  atom$adp_start
  atom$adp_eid_string <M0039813>
  atom$adp_no_more_chunks
➪ atom$adp_end
```


atom$adp_more_chunks
14 ($0E)

Description

atom$adp_more_chunks tells the client the host is ready for the next data chunk in an asynchronous data transfer session.

Syntax

```
atom$adp_more_chunks
```

Return Value

None.

Example

The following host-to-client example tells the client the host is ready for the next data chunk:

```
  atom$adp_start
  atom$adp_eid <1>
➪ atom$adp_more_chunks
  atom$adp_end
```

atom$adp_no_more_chunks
9 ($09)

Description

atom$adp_no_more_chunks signals to the receiver that the data transmission is complete.

Syntax

```
atom$adp_no_more_chunks
```

Return Value

None.

Examples

The following client-to-host example tells the host the data transfer session is complete:

```
  atom$adp_start
  atom$adp_eid <1>
➪ atom$adp_no_more_chunks
  atom$adp_end
```

The following host-to-client example tells the client the data transfer session is complete:

```
  atom$adp_start
  atom$adp_eid_string <M0039813>
➪ atom$adp_no_more_chunks
  atom$adp_end
```


atom$adp_route_tkn
4 ($04)

Description

atom$adp_route_tkn defines a host server routing token the client should use regarding the entire data transfer session.

Syntax

```
atom$adp_route_tkn <word>
```

`<word>`	Specifies a hex value (2 bytes) that defines a host server routing token the client should use regarding this session. The value is a code of 2 ASCII characters. For example, the value 4Ex 57x defines a routing token of NW.

Return Value

None.

Example

The following host-to-client example defines a host server routing token the client should use regarding this session:

```
atom$adp_start
atom$adp_transaction_type <2>
atom$adp_eid_string <M0039813>
➪ atom$adp_route_tkn <4Ex 57x>
atom$adp_abort_tkn <4Ex 57x>
atom$adp_end
```

atom$adp_start
1 ($01)

Description

atom$adp_start marks the beginning of an ADP code stream that may start, continue, or stop an asynchronous data transfer session. Note that each ADP code stream must be terminated with **atom$adp_end**.

Syntax

```
atom$adp_start
```

Return Value

None.

Examples

The following examples show a set of ADP code streams that illustrate the interactive asynchronous protocol for an upload session with the host. Each code stream must be started with **atom$adp_start**.

The following host-to-client example starts an ADP code stream that establishes an asynchronous data transfer session with the client:

```
➪ atom$adp_start
  atom$adp_transaction_type <1>
  atom$adp_eid <1>
  atom$adp_chunk_size <1900>
  atom$adp_chunk_count <1>
  atom$adp_route_tkn <4Ex 57x>
  atom$adp_abort_tkn <4Ex 57x>
  atom$adp_end
```

The following client-to-host example starts an ADP code stream that sends data to the host:

```
➪ atom$adp_start
  atom$adp_eid <1>
  atom$adp_data <00x 01x C3x 1Ax 35x 00x 02x … >
  atom$adp_end
```

The following host-to-client example starts an ADP code stream that tells the client the host is ready for more data:


```
atom$adp_start
atom$adp_eid <1>
atom$adp_more_chunks
atom$adp_end
```

The following client-to-host example starts an ADP code stream that tells the host the data transfer session is complete:

```
atom$adp_start
atom$adp_eid <1>
atom$adp_no_more_chunks
atom$adp_end
```

atom$adp_transaction_type
15 ($0F)

Description

atom$adp_transaction_type defines whether the data transfer is an upload or a download session.

Syntax

```
atom$adp_tranaction_type <dword>
```

`<dword>` Specifies the transaction ID. Values are:

`1`	Upload transaction
`2`	Download transaction

Return Value

None.

Examples

The following host-to-client example defines the data transfer as a download session:

```
  atom$adp_start
➪ atom$adp_transaction_type <2>
  atom$adp_eid_string <M0039813>
  atom$adp_route_tkn <4Ex 57x>
  atom$adp_abort_tkn <4Ex 57x>
  atom$adp_end
```

The following host-to-client example defines the data transfer as an upload session:

```
  atom$adp_start
➪ atom$adp_transaction_type <1>
  atom$adp_eid <1>
  atom$adp_chunk_size <1900>
  atom$adp_chunk_count <1>
  atom$adp_route_tkn <4Ex 57x>
  atom$adp_abort_tkn <4Ex 57x>
  atom$adp_end
```

