Conventions

This document covers the following topics:

- Control Statement Syntax
- Syntax Conventions
- Control Statement Rules
- Parameter Values

Notation *vrs*, *vr*, or *v*: When used in this documentation, the notation *vrs* or *vr* stands for the relevant version of a product. For further information on product versions, see *version* in the *Glossary*.

Control Statement Syntax

Utility control statements have the following format:

```
utility  function  parameter-list
```

where

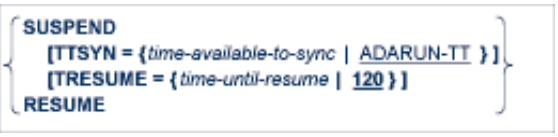
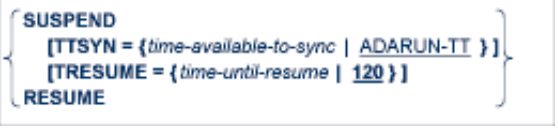
utility	is the name of the utility to be executed. Examples of utility names include: ADAORD ADADBS ADAINV
function	is the name of the specific utility operation to be executed. For example: ADAORD REORDATA ADADBS ADD ADAINV COUPLE Most single-function utilities (ADASEL, ADAULD, etc.) whose function is implicit have either no function value or an optional one.
parameter-list	is a list of parameters following the function. Parameters in the list are almost always keywords with the format: *parameter=value* A parameter may have one or more operands, and keyword parameters may be specified in any order. Most parameters require that you select or otherwise specify an operand value. Some operands are positional (value1 , value2 ,..., valuex), meaning that the values must be in a certain order as described in the text. All parameters must be separated by commas. In the statement syntax descriptions in this documentation, parameters are listed vertically (stacked) or are separated by vertical bars (|). Each list shows all possible parameters, from which one or more can (or must) be specified. Although parameters in the list must be separated by commas, these commas are omitted in the syntax statements when the parameters are stacked.

Syntax Conventions

The following table describes the conventions used in syntax diagrams of Adabas statements.

Convention	Description	Example
uppercase, bold	Syntax elements appearing in uppercase and bold font are Adabas keywords. When specified, these keywords must be entered exactly as shown.	**ADADBS CHANGE FILE =** *file-number* The syntax elements ADADBS, CHANGE, and FILE are Adabas keywords.

Convention	Description	Example
lowercase, italic, normal font	Syntax elements appearing in lowercase and normal, italic font identify items that you must supply.	**ADADBS CHANGE FILE =** *file-number* The syntax element *file-number* identifies and describes the kind of value you must supply. In this instance, you must supply the number of the file affected by the ADADBS CHANGE operation.
mixed case, normal font	Syntax elements appearing in mixed case and normal font (not bold or italic) identify items established by other Adabas control statements. This notation is usually used to identify how default values are determined for some parameters in Adabas syntax.	**[SORTDEV = {** *device-type* **|** ADARUN-device **}]** The syntax element "ADARUN-device" indicates that the device type identified by the ADARUN DEVICE parameter will be used if a different device type is not specified. The literal "ADARUN-device" should *not* be specified for the SORTDEV parameter.
underlining	Underlining is used for two purposes: 1. To identify default values, wherever appropriate. Otherwise, the defaults are explained in the accompanying parameter descriptions. 2. To identify the short form of a keyword.	**[LRECL = {***record-buffer-length* **|** 4000 **}]** In the example above, 4000 is the default that will be used for the LRECL parameter if no other record buffer length is specified. **DEVICE** In the example above, the short version of the DEVICE parameter is DE.
vertical bars (|)	Vertical bars are used to separate mutually exclusive choices. **Note:** In more complex syntax involving the use of large brackets or braces, mutually exclusive choices are stacked instead.	**ADAORD { RESTRUCTUREF | REF }** In the example above, you must select RESTRUCTUREF or REF for this ADAORD function. There are no defaults.

Convention	Description	Example
brackets ([ ])	Brackets are used to identify optional elements. When multiple elements are stacked or separated by vertical bars within brackets, only one of the elements may be supplied.	 In this example, the SORTSEQ parameter and the MU, NU, and STARTISN subparameters are optional. **Note:** Note that the mutually exclusive choices for the SORTSEQ parameter are stacked.
braces ({ })	Braces are used to identify required elements. When multiple elements are stacked or separated by vertical bars within braces, one and only one of the elements must be supplied.	 In this example, either the SUSPEND or RESUME parameter is required.
indentation	Indentation is used to identify subparameters of a parameter.	 In this example, TTSYN and TRESUME are subparameters of the SUSPEND parameter.
ellipsis (...)	Ellipses are used to identify elements that can be repeated. If the term preceding the ellipsis is an expression enclosed in square brackets or braces, the ellipsis applies to the entire bracketed expression.	 In this example, the FIELD parameter can be repeated. In addition, more than one option can be associated with a field.
other punctuation and symbols	All other punctuation and symbols must be entered exactly as shown.	 In this example, the single quotation marks must be specified around the field definitions and their associated options. In addition, options must be separated by commas.

Control Statement Rules

The following rules apply for the construction of utility control statements:

1. Each control statement must contain a utility name in positions one through six.

2. The utility function name follows the utility name, separated by at least one space.
3. Keyword parameter entries and multiple values within keyword entries must be separated by commas.
4. No space is permitted before or after an equals symbol (=).
5. The comma following the last parameter entry of a statement is optional.
6. Control statement processing ends with position 72 or when a space is encountered after the beginning of the parameter list. Entries made in positions 73-80 are not processed.
7. A statement that contains an asterisk (*) in position one is read as a comment and is not processed.
8. Control statements are continued by specifying the extra parameters on a new statement following (and separated by at least one space from) the utility name in positions one through 6.

Parameter Values

Variable values actually specified following the equals symbol (=) in parameters (represented by italicized labels in the preceding examples and elsewhere in this documentation) have the following syntax:

```
parameter = value
parameter = value-list
parameter = value-range
```

where *value* is as described in the following sections. Parameters *value-list* and *value-range* are variations of *value*, and are allowed either in place of or with *value*, depending on the individual parameter rules as described in the text.

value

The *value* parameter may consist of a number or a string of alphanumeric or hexadecimal characters. In some optional keyword parameters, a default value is assumed if the parameter is not specified.

Alphanumeric Values

Alphanumeric values are specified in one of the following ways:

If the value comprises . . .	Apostrophes around it are . . .
only upper- or lowercase letters, numeric digits and minus (-)	optional
any other characters including an apostrophe itself (which must be entered twice)	required

Numeric Values

Numeric values are specified as follows:

If the value represents . . .	Specify . . .
a number of either blocks or cylinders	the letter B must immediately follow the value if blocks are being specified; otherwise, cylinders are assumed: SIZE=200B (200 blocks) SIZE=200 (200 cylinders)
an Adabas file	a one- to four-digit number (leading zeros permitted): FILE=3 FILE=03 FILE=162
a device type	a four-digit number corresponding to the model number of the device type to be used: DEVICE=3380
a field name or descriptor	a two-character field name corresponding to the field name or descriptor: FIELD1=NA

Hexadecimal values are accepted if this is specified in the parameter description. Hexadecimal values must be within apostrophes following the indicator X:

```
X’0002DC9F’
```

value-list

value,... (numeric values)

```
BITRANGE=2,10,2
```

or

’value,...’ (alphanumeric values)

```
UQDE=’AA,AC,AE’
```

value-range

value - value, ...

```
ISN=600-900,1000-1200
```

Individual values within a value list or value range may be positional if they relate to values specified on corresponding parameters. For example:

```
ADADBS  UNCOUPLE  FILES=13,20,PASSWORD='PW13,PW20'
```

-instructs the ADADBS UNCOUPLE function to uncouple files 13 and 20, which are password-protected.

The passwords (specified by the PASSWORD parameter) must be in the same order as their corresponding files in the FILES parameter.

If file 13 is *not* password-protected, either the PASSWORD parameter must be specified with a placeholder comma as shown below

```
... PASSWORD=',PW20'
```

-to position the password "PW20" to the corresponding position of file 20 in the FILES value list, or FILES must specify file 20 first.