Criteria Syntax Summary

Contexts: Criteria Run-time Environments

Proposed Cross Section
Design & Computation Manager

Generic Expressions:	Example:
<numeric constant>	12.345
<user defined variable>	"THICKNESS"
<define_dgn variable>	"EOP in DGN"
<string>	ABC123
<double-precision variable>	_d_<*string*>
<string variable>	_s_<*string*>
<marked point name>	*_mp_*<*string*>
<selection set>	*_ss_*<*string*>
<DGN or D&C element>	*_ed_*<*string*>

Comments:

/*…any text…*/

/*
…any text…
*/

Logical Operators:

AND OR NOT

Relational Operators:

> >= < <= =

Arithmetic Operators:

+ - * / % (modulo) ! (factorial) @ *(exponentiation)*

Delimiter Operators:

, () [] {} ^ "

Assignment Operator:

= <<

Built-In Math Functions:

SIN SINH ASIN COS COSH ACOS TAN TANH ATAN LOG LOG10 SQRT ABS EXP INT

Intrinsic Variables – Numeric and String

All Contexts

JOB NUMBER
(Automatically assigned by Proposed Cross Section Context and Design & Computation Manager Compute Mode > Comp Book)

CHAIN NAME
(Automatically assigned by Proposed Cross Section Context and Design & Computation Manager Compute Mode > Comp Book)

END OF FILE
(Used when reading from an ASCII file)

Proposed XS Contexts

FIRST XS
(TRUE only for the first section in the traditional Proposed XS context. In any other situation it's FALSE)

DOWNSTREAM XS STATION *(next station)*
DOWNSTREAM XS REGION *(next region)*

Determined by Superelevation Shapes:

NUMBER OF PAVEMENT POINTS
MEDIAN WIDTH
DELTA PAVEMENT ELEVATION
PAVEMENT SLOPE

CLUSTER BASELINE
CLUSTER PROFILE
CLUSTER PGL CHAIN
CLUSTER TIE
CLUSTER OFFSET

** note: the above 5 variables (CLUSTER *) can only be used if they are first assigned to a redefinable string variable. For example,*

_s_ClusterBaseline = Cluster Baseline
if _s_ClusterBaseline = ^RampA^ then

Determined by XS Cell

HORIZONTAL TO VERTICAL SCALE RATIO
HORIZONTAL SCALE

** note: the above 2 variables (HORIZONTAL *) can only be used if they are first assigned to a redefinable numeric variable. For example,*

_d_Ratio = HORIZONTAL TO VERTICAL SCALE RATIO
if (_d_Ratio = 2) then

Determined by XS Cell Chain

ARC RADIUS
STATION
REGION
SKEW ANGLE FOR CLUSTER
SKEW ANGLE FOR PATTERN

Determined from User-Supplied Chain Names

CHAIN <chain name> STATION
CHAIN <chain name> REGION

Determined by Existing Ground

ELEVATION DIFFERENCE

Determined by Drainage Components

NEAREST DRAINAGE COMPONENT DISTANCE
Drainage Node ID
Drainage Node Description
Drainage Node Type
Drainage Node Library Item Name
Drainage Node Library Item Description
Drainage Node Reference Chain
Drainage Node Reference PGL
Drainage Node Station
Drainage Node Offset
Drainage Node Location X
Drainage Node Location Y
Drainage Node Rotation
Drainage Node Alignment Azimuth
Drainage Node Elevation
Drainage Node Reference Elevation
Drainage Node Cumulative Discharge
Drainage Node Cumulative Area
Drainage Node Cumulative C Value
Drainage Node Cumulative Intensity
Drainage Node Hyd Center X
Drainage Node Hyd Center Y
Drainage Node Elev Point X
Drainage Node Elev Point Y
Drainage Node Origin X
Drainage Node Origin Y
Drainage Node Minimum Depth
Drainage Node Maximum Depth
Drainage Node Minimum Invert Elevation
Drainage Node Maximum Rise In
Drainage Node Maximum Rise Out
Drainage Node Minimum Invert Elevation

DRAINAGE BOTTOM FRONT POINT
DRAINAGE BOTTOM BACK POINT

Drainage Headwall ID
Drainage Headwall Description
Drainage Headwall Type
Drainage Headwall Library Item Name

Drainage Headwall Library Item Description
Drainage Headwall Reference Chain
Drainage Headwall Reference PGL
Drainage Headwall Station
Drainage Headwall Offset
Drainage Headwall Location X
Drainage Headwall Location Y
Drainage Headwall Rotation
Drainage Headwall Alignment Azimuth
Drainage Headwall Elevation
Drainage Headwall Reference Elevation
Drainage Headwall Cumulative Discharge
Drainage Headwall Cumulative Area
Drainage Headwall Cumulative C Value
Drainage Headwall Cumulative Intensity
Drainage Headwall Hyd Center X
Drainage Headwall Hyd Center Y
Drainage Headwall Elev Point X
Drainage Headwall Elev Point Y
Drainage Headwall Origin X
Drainage Headwall Origin Y
Drainage Headwall Minimum Depth
Drainage Headwall Maximum Depth
Drainage Headwall Minimum Invert Elevation

Drainage Link ID
Drainage Link Description
Drainage Link Type
Drainage Link Upstream Node
Drainage Link Downstream Node
Drainage Link Shape
Drainage Link Material
Drainage Link Corr Type
Drainage Link Library Item
Drainage Link Number of Barrels
Drainage Link Actual Length
Drainage Link Hydraulic Length
Drainage Link Manning's N Value
Drainage Link Slope
Drainage Link Rise
Drainage Link Span
Drainage Link Discharge
Drainage Link Soffit Upstream
Drainage Link Soffit Downstream
Drainage Link Invert Upstream
Drainage Link Invert Downstream
Drainage Link Begin Point
Drainage Link End Point

Drainage Component Front Point
Drainage Component Back Point

Intrinsic Variables - D&C Compute Context

D&C ITEM NAME
D&C ITEM DESCRIPTION

Intrinsic Variables – Profile Sub

Profile Sub Chain
Profile Sub Horizontal Scale
Profile Sub Vertical Scale
Profile Sub Reference DP X
Profile Sub Reference DP Y
Profile Sub Reference DP Station
Profile Sub Reference DP Region
Profile Sub Reference DP Elevation

The above variables take their values from either:
a) The parameters explicitly specified in the sub call
b) The Profile Cell in the dgn file

Mark Points

Intrinsic Mark Points

Pn (where n=1, 2,… 20)
P[<arithmetic expression>]

XS EG LT PT *intrinsic marked point for left end of existing ground*
example: draw skip to XS EG LT PT

XS EG RT PT *intrinsic marked point for right end of existing ground*
example: draw skip to XS EG RT PT

XS HI PT *intrinsic marked point for upper right edge of cross section cell search boundary*
example: draw skip to XS HI PT

XS LO PT *intrinsic marked point for lower left edge of cross section cell search boundary*
example: draw skip to XS LO PT

XS BL PT *intrinsic marked point for X and Y axis intersection of cross section cell*
example: draw skip to XS BL PT

Non-Intrinsic Mark Points

Xn Yn
x[<arithmetic expression>] y[<arithmetic expression>]
<mark point name>

Non-Intrinsic Mark Point Variables:

Xn (where n=1, 2,… 1000)	Yn (where n=1, 2,… 1000)
X[<arithmetic expression>]	Y[<arithmetic expression>]

XS View:
Starting Location: p1

<mark point name>.x	*Automatically assigned by 3PC*
<mark point name>.y	*Automatically assigned by 3PC*
<mark point name>.elev	*Automatically assigned by 3PC*
<mark point name>.distalongchain	*Automatically assigned by 3PC*
<mark pointname>.off	*Automatically assigned by 3PC*

Plan View:
Starting Location: global origin

<mark point name>.x	*Automatically assigned by 3PC*
<mark point name>.y	*Automatically assigned by 3PC*
<mark point name>.elev	*Undefined (Set to huge number)*
<mark point name>.distalongchain	*Undefined (Set to huge number)*
<mark pointname>.off	*Undefined (Set to huge number)*

Profile View:
Starting Location: Reference DP

<mark point name>.x	*Automatically assigned by 3PC*
<mark point name>.y	*Automatically assigned by 3PC*
<mark point name>.elev	*Set relative to reference elev*
<mark point name>.distalongchain	*Set relative to reference distAlongChain*
<mark pointname>.off	*Set to 0*

NOTE: 3 Port Mark Point Rules:

- *Mark Point Name in XS: Point can be used in Plan and Profile Views.*
- *Mark Point Name in Plan: Point can be used in Profile View if < mark point name>.distAlongChain and < mark point name>.elev assigned. Point cannot be used in XS View.*
- *Mark Point Name in Profile: Point can be used in Plan View only. Point cannot be used in XS View.*
  - *< mark point name>.elev = (Ref DP elev) + (Point's Prof View Y - Ref DP Prof View Y) * vert scale / horiz scale*
  - *< mark point name>.distAlongChain = (Ref DP distAlongChain) + (Point's Prof View X – Ref DP Prof View X) –Gaps, where Gaps = the sum of the lengths of all the gaps between the Point and the Ref DP.*

Assignment Statements for Mark Points:

< mark point name> = <mark point name>
< mark point name>.x = <arithmetic expression>
< mark point name>.y = <arithmetic expression>
< mark point name>.elev = <arithmetic expression>
< mark point name>.distAlongChain = <arithmetic expression>

Note: < mark point name> must be marked prior to performing any assignments

If-Then expressions:

```
IF <conditional expression> THEN
  {

  }
ELSE IF <conditional expression> THEN
  {

  }
ELSE
  {

  }
```

Logical Intrinsic Variables (Used Only in If-Then Statements):

Proposed XS Context

MATERIAL BELOW = <Define name>
MATERIAL ABOVE = <Define name>
MATERIAL BELOW Xn Yn
STATION WITHIN PROFILE = <profile name>
CHAIN <_s_string> STATION WITHIN PROFILE = <profile name>
INTERSECT DGN = <Define DGN name>

While expressions:

WHILE <conditional expression>
{

}

BREAK
(Used only in a WHILE loop, terminates the current loop.)

CONTINUE
(Used only in a WHILE loop, skips all remaining lines within the current WHILE loop and does a new iteration of the loop.)

Draw Statements:

DRAW [skip] [GG=<gg name>]<slope component> <distance component>

<gg name>:
^<string>^
_s_<string>

<slope component>:
dx = <arithmetic expression> dy = <arithmetic expression>
slope = <slope expression>
rise:run = <arithmetic expression>:<arithmetic expression>
run:rise = <arithmetic expression>:<arithmetic expression>
angle = <arithmetic expression>

<slope expression>:
<arithmetic expression>
<arithmetic expression>:<arithmetic expression>

<distance component>:
FOR dx = <arithmetic expression>
FOR dy = <arithmetic expression>
TO <marked point coordinate> *(Ex: TO X1, TO Y33, etc.)*
TO Existing Ground
TO <define_dgn variable> [WITHIN DISTANCE = <arithmetic expression>]
TO XS <define_dgn variable> [WITHIN DISTANCE = <arithmetic expression>]
TO CHAIN <chain name>
TO PROFILE ELEVATION = <prof name>[,<prof name>,…]
TO CHAIN <chain name> PROFILE ELEVATION = <prof name>[,<prof name>,…]
TO INTERSECTION <marked point> <marked point>
FOR DISTANCE = <arithmetic expression>

DRAW [skip] [GG=<gg name>]<all-inclusive slope and distance component>

<all-inclusive slope and distance component>:
dx = <arithmetic expression> dy = <arithmetic expression>
TO <marked point>
TO CHAIN <chain name> PROFILE ELEVATION = <prof name>[,<prof name>,…]
TO <drainage point>
TO CHAIN <chain name> DISTALONGCHAIN= <arithmetic expression> OFF = <arithmetic expression>
TO CHAIN <chain name> DISTALONGCHAIN = <arithmetic expression> ELEV= <arithmetic expression>

<drainage point>:
DRAINAGE COMPONENT FRONT POINT
DRAINAGE COMPONENT BACK POINT
DRAINAGE LINK BEGIN POINT
DRAINAGE LINK END POINT

<marked point>:
Pn
P[<arithmetic expression>]
Xn Yn
X[<arithmetic expression>] Y[<arithmetic expression>]
_mp_<string>

DRAW [skip] [GG=<gg name>]TRACE [<define_dgn variable>]
<direction>
[OFFSET=<arithmetic expression>]
<distance component>

<direction>:
IN
OUT

<distance component>:<distance component> TO ENDPOINT (only for draw trace “material”)

DRAW *[GG=<gg name>]*TEXT CHAR = ^<string>^ <plot parameters>
DRAW *[GG=<gg name>]*TEXT VALUE = <expression> <plot parameters>

DRAW *[GG=<gg name>]*CELL = <cell name> *[CELL LIBRARY=<library name>]* <plot parameters>
*Note: When cell name is composed of multiple words, the cell name MUST be encompassed within carets. i.e. draw cell = ^raised pavement marker^*

DRAW *[GG=<gg name>]*<type> <plot parameters>

<type>:
STATION TEXT
ELEVATION TEXT
DISTANCE TEXT BETWEEN Xi Yi AND Xj Yj
SLOPE TEXT BETWEEN Xi Yi AND Xj Yj

<plot parameters>:
NUMBER OF DECIMAL PLACES = <arithmetic expression>
JUSTIFICATION = <justification>
ANGLE = <angle definition>
FORMAT = <slope format>

CO = <arithmetic expression> or <bylevel>
LV = <arithmetic expression>
CLASS = Primary or Construction
LVNAME = <string>
*NOTE: Wildcards are supported for Level Names. i.e. lvname = patt**
WT = <arithmetic expression> or <bylevel>
LC = <arithmetic expression> or <bylevel>
adHocNumericValue(<string expression>) = <arithmetic expression>
adHocStringValue(<string expression>) = ^<string expression>^
adHocQuantityValue(<string expression>) = < arithmetic expression>
adHocUnitValue(<string expression>) = ^<string expression>^
adHocRemarksValue(<string expression>) = ^<string expression>^
FT = <arithmetic expression>
TTFTNAME = <string>
SHXFTNAME = <string>
OFFSET = <arithmetic expression>
SCALE = <arithmetic expression>
TH = <arithmetic expression>
TW = <arithmetic expression>
VERTICAL DIMENSION LINE LENGTH = <arithmetic expression>
XS = <arithmetic expression>
YS = <arithmetic expression>

<angle definition>:
LINE BETWEEN Xi Yi AND Xj Yj
<arithmetic expression>

<slope format>:
/
:
%
RISE:RUN
RUN:RISE
RUN:1
1:RUN
FT/FT
M/M

<justification>:
LT, LC, LB, CT, CC, CB, RT, RC, RB

Set:

SET JOB NUMBER <string> *(D&C Context Only)*
SET PLOT PARAMETERS <plot parameters>
Note: < plot parameters> includes the following:
LV =
Co =
Wt =
LC =
Class = Primary or Construction
adHocNumericValue(<string expression>) = <arithmetic expression>
adHocStringValue(<string expression>) = ^<string expression>^

adHocQuantityValue(<string expression>) = < arithmetic expression>
adHocUnitValue(<string expression>) = ^<string expression>^
adHocRemarksValue(<string expression>) = ^<string expression>^
SET RISE:RUN
SET RUN:RISE
SET PLACE INFLUENCE ON
Note: If "set place influence on" command is issued, then any element drawn will be drawn using the Symbology of the ddb item.
SET PLACE INFLUENCE OFF
SET SKEW ANGLE _d_SkewAngle
SET D&C QUANTITY [<double-precision expression> or <string variable>]
SET D&C ITEM NAME <string variable>
SET D&C ITEM DESCRIPTION <string variable>
SET D&C QUANTITY DESCRIPTION <string>
SET D&C QUANTITY EXTENDED DESCRIPTION <string>
SET D&C QUANTITY REMARKS <string>

COGO Functions:

<double-precision variable> = cogo_Distance_Between_Points (<marked point name>, <marked point name >)
<double-precision variable> = cogo_Angle_Between_Points (<marked point name >, <marked point name >)
<double-precision variable> = cogo_DistAlongChain (<marked point name >, <chain name>)
<double-precision variable> = cogo_OffsetFromChain (<marked point name >, <chain name>)
<double-precision variable> = cogo_TangentAngleChain (<marked point name >, <chain name>)
<double-precision variable> = cogo_ProfElev(_s_sta,_s_reg,_s_prof)

NOTE: For the last three functions, if the marked point name is outside of the range of the alignment the function will return a huge number.

Conversion Functions:

<string variable> = CVT_D_S0(<arithmetic expression>)
<string variable> = CVT_D_S4(<arithmetic expression>)
<double-precision variable> = CVT_S_D(<string variable>)

<string variable> = CVT_D_S0_ADMS(<arithmetic expression>)
<string variable> = CVT_D_S4_ADMS(<arithmetic expression>)
<string variable> = CVT_D_S0_ADM(<arithmetic expression>)
<string variable> = CVT_D_S4_AD(<arithmetic expression>)

<arithmetic expression> = CVT_StaReg_distAlongChain (<string variable>, <string variable>, <chain name>)
<string variable> = CVT_D_STA(<arithmetic expression>)
<double-precision variable> = CVT_STA_D(<string variable>)

<string variable> = CVT_distAlongChain_Sta (<arithmetic expression>, <chain name>)
<string variable> = CVT_distAlongChain_Reg (<arithmetic expression>, <chain name>)
NOTE: Using a very large <arithmetic expression> will return the end station for the chain. Of course, the beginning station can be obtained by setting the <arithmetic expression>=0.0.

NOTE: Additional options have been added to the conversion function "CVT_distAlongChain_Sta" to allow the user to individually control the format of the stationing. Number of decimal places and station format are now optional. The following examples illustrate these optional enhancements.
Example 1 specifies only number of decimals:
_d_stadec = 3 → use a valid integer
_s_sta = cvt_distalongchain_sta (_mp_clpt.distalongchain, _s_chain_name, _d_stadec)

Example 2 specifies only station format:
_s_stafmt = ^1+234^ or ^12+34^ or ^1234^ or ^+234^ or ^+34^ or ^1^ or ^12^
_s_sta = cvt_distalongchain_sta (_mp_clpt.distalongchain, _s_chain_name, DEFAULT, _s_stafmt)

Example 3 specifies both number of decimals and station format:
_d_stadec = 3 → use a valid integer
_s_stafmt = ^1+234^ or ^12+34^ or ^1234^ or ^+234^ or ^+34^ or ^1^ or ^12^
_s_sta = cvt_distalongchain_sta (_mp_clpt.distalongchain, _s_chain_name, _d_stadec, _s_stafmt)

String Parsing:

<string variable> = <string variable>[N:n]

Note: Go to the "N"th character and grab the next "n" characters:
Example:

*_s_fulltext = ^0123456789^*

_s_subtext = _s_fulltext[4:3]

*the above syntax would make _s_subtext = ^345^*

DTM Statements:

<double-precision variable> = dtm_elevation(<TIN File Name>,<marked point name>)
NOTE: Returns huge number if marked point is outside the TIN hull.

ASCII File Commands:

ASCII OPEN [READONLY] FILE = <file name>
ASCII APPEND FILE = <file name>
ASCII CLOSE FILE = <file name>
ASCII WRITE FILE = D&C
FORMAT = ITEM TABLE
ITEM REPORT
ASCII WRITE FILE = <file name>
FORMAT = ^<C printf format>^
VAR = <variable list>

<file name>:
<string>
<string variable>
<variable list>:
<string variable>,<double-precision variable>

ASCII READ FILE = <file name>
*FORMAT = ^<C scanf format>^*
VAR = <variable list>

<file name>:
<string>
<string variable>
<variable list>:
<string variable>,<double-precision variable>

Prompts:

PROMPT <string label> VARIABLE = <string variable> MODE = FILE SELECT [FILTER=<file filter>]
<file filter>:
[<string>][*][<string>][.][<string>][*][<string>]

PROMPT <string label> VARIABLE = <marked point name> MODE = DP

PROMPT <string label> MODE = MESSAGE

PROMPT <string label> VARIABLE = <receptor> MODE = OPTION
CONTENT = <option name>, <option name>

PROMPT <string label> VARIABLE = <receptor> MODE = KEYIN
<string label>:
<string variable>
^<string>^
<option name>:
<string>
<receptor>:
<double-precision variable>
<string variable>

Prompts Intrinsic Variables:

if (OK) then…

if (CANCEL) then…

Note that the former condition is true if a prompt command has been processed and the OK push button was pressed. The latter condition is true if a prompt command has been processed and the CANCEL push button was pressed.

EXIT Terminates the Criteria Process

Subroutines:

SUBROUTINE [XS, PLAN or PROFILE] <sub name>
{
/* criteria statements */
}
END SUBROUTINE

CALL <sub name>

CALL <sub name> (PLAN VIEW [=<dgn file name>])

CALL <sub name> (PROFILE VIEW [=<dgn file name>]
X=<arithmetic expression>
Y=<arithmetic expression>
HORIZONTAL SCALE=<arithmetic expression>
VERTICAL SCALE=< arithmetic expression>
STA=<string>
REG=< arithmetic expression>
ELEV=< arithmetic expression>
GAP=<YES or NO>,
CHAIN = <chain name>)

Profile Subroutine Call using the Profile Cell:
CALL <sub name> (PROFILE VIEW [=<dgn file name>]
CHAIN = <chain name>)

<sub name>:
<alphanumeric string>

<dgn file name>:
<alphanumeric string>

MicroStation Commands:

SEND MICROSTATION <string>

Cogo Commands:

SEND COGO <string>

Selection Sets

General Element Assignment:

_ed_<string> = _ss_<string>([_ed_<string>+]<arithmetic expression>)
_ed_<string> = _ed_<string>
_ss_<string> = _ss_<string> - _ed_<string>

Input Source:

_ss_<string> = DGN SELECTION SET
_ss_<string> = D&C SELECTION SET

Input Intrinsic Variables:

_d_<string> = DGN NUMBER OF ELEMENTS
_d_<string> = D&C NUMBER OF ELEMENTS

Element Variables Read Only:

_ed_<string>.x1
_ed_<string>.y1
_ed_<string>.z1
_ed_<string>.x2
_ed_<string>.y2
_ed_<string>.z2
_ed_<string>.type = <TEXT, CELL, LINE, LINE_STRING,ARC, CURVE, COMPLEX_SHAPE, COMPLEX_STRING,ELLIPSE>
_ed_<string>.length
_ed_<string>.area
_ed_<string>.radius
_ed_<string>.delta
_ed_<string>.text * this is the value of the text string
_ed_<string>.adHocNumber
_ed_<string>.level
_ed_<string>.lvname
_ed_<string>.color
_ed_<string>.weight
_ed_<string>.style
_ed_<string>.UserDefinedLineStyle
_ed_<string>.cellName
_ed_<string>.complexComponentElement

Element Variables Read and Assign:

_ed_<string>.adHocNumericValue(<string expression>) = (<arithmetic expression>)
_ed_<string>.adHocStringValue(<string expression >) = (<string expression>)
_ed_<string>.adHocQuantityValue(<string expression>) = (< arithmetic expression>)
_ed_<string>.adHocUnitValue(<string expression>) = (<string expression>)
_ed_<string>.adHocRemarksValue(<string expression>) = (<string expression>)

_ed_<string>.dcAttribute = (<string expression>)
_ed_<string>.UserDefinedLineStyle = (<string expression>)

Update Elements in DGN File:

WRITE AdHoc(_ed_<string>)

Diagnostic Command available for input files

DIAGNOSTIC PRINT SHAPE DATA (Proposed XS Input File only)
DIAGNOSTIC PRINT EXISTING GROUND (Proposed XS Input File only)
DIAGNOSTIC PRINT PATTERN-SHAPE INTERSECTIONS (Proposed XS Input File only)
DIAGNOSTIC PRINT DRAINAGE INFORMATION (Proposed XS Input File only)
DIAGNOSTIC PRINT SMALL ELEMENTS (Proposed XS & Earthwork Input Files)