<New Debugger>

1. Overview

디버깅의 버그(bug)는 벌레를 뜻하며, 디버그(debug)는 '해충을 잡다'라는 뜻이다. 프로그램의 오류를 벌레에 비유하여 오류를 찾아 수정하는 일이라는 의미로 쓰인다. 처음에 컴퓨터가 발명되었을 때는 그 크기가 집채만하였다. 정상적으로 작동하던 프로그램이 갑자기 오류가 발생하였는데, 그 오류의 원인을 찾아보니 나방(bug) 한 마리가 전선 사이에 끼여서 합선을 일으킨 것에서 유래가 되었다.

ABAP 은 COBOL 에서 파생된 언어이며, 절차적인 구조를 기본으로 하기 때문에 디버깅이 더 강력한 기능을 수행한다(객체 지형 프로그래밍 및 Web Dynpro for ABAP 에서도 ABAP Debugger 를 사용한다.). ABAP 이 타 언어에 비해 쉽게 느껴지는 이유중의 하나가 ABAP 디버거의 탁월한 기능과 편리한 GUI 때문이라고 생각한다.(사실 ABAP 은 지속적으로 개발되고 있기 때문에 우리가 알고 있는 것은 빙산의 일각일 뿐이다. ABAP 이 쉽다고 말하는 사람들의 말을 절대 맹신하지 말자. 아는 만큼 보이는 것이다.)

또한, 대부분의 타 언어에서는 디버깅을 실행하기 위해 독립 프로그램을 실행하여야 하지만 ABAP 은 통합된 환경으로 에디터와 디버깅 화면을 자유롭게 변환할 수 있는 장점이 있다. ABAP Debugger 는 Classical debugger 와 이후의 New debugger 가 있다. New debugger 는 자체 디버깅 모드가 존재 하는 등 Classic 모드보다 많은 기능을 제공한다.

ABAP Debugger 는 ABAP Workbench 에 통합된 TOOL 이다.

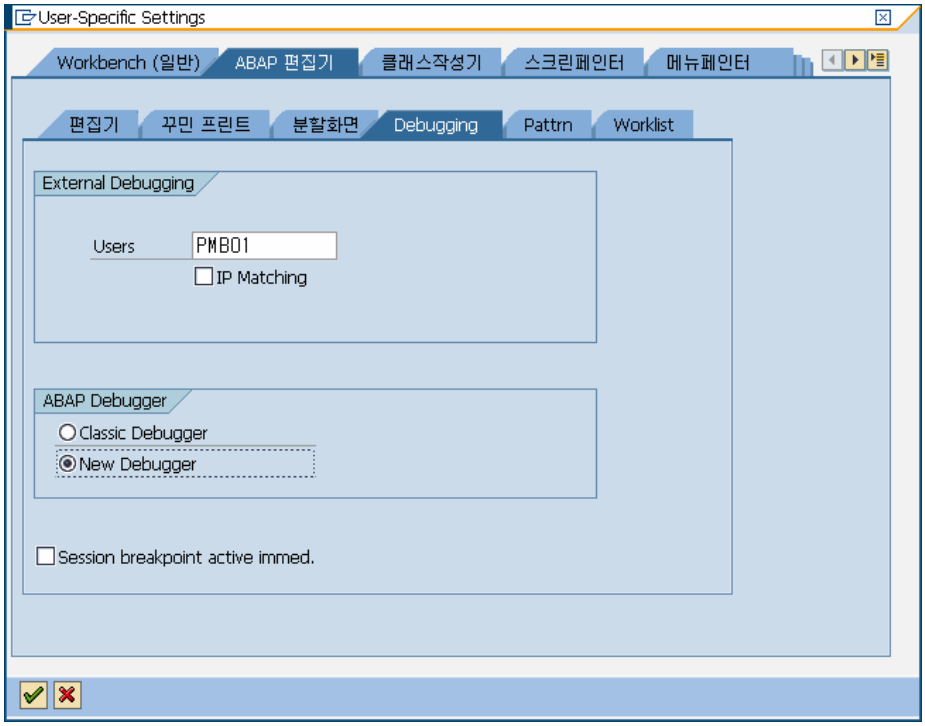
New debugger 는 기능상으로 보완된 것들이 대부분이기 때문에, Classical 디버깅만 제대로 이해하고 활용할 수 있다면 New debugger 는 쉽게 습득할 수 있다.

Classical Debugger
- ABAP 프로그램 실행시 동일 세션에서 열린다.
- Conversion Exit 과 같은 일부 ABAP 프로그램은 디버깅할 수 없는 제약이 있다.

New Debugger
- ABAP 프로그램과 별개의 외부세션에서 열린다.
- ABAP 프로그램 종류에 관계없이 디버깅을 수행할 수 있다.

[그림 1]과 같이 Rel 6.40 버전 이후부터는 Classic Debugger 와 New Debugger 를 선택할 수 있다. T-CODE:SE38 ABAP 에디터에서 메뉴 [Utilities] → [Settings]를 선택하자. Rel 7.00 버전부터는 New Debugger 가 기본으로 설정된다.

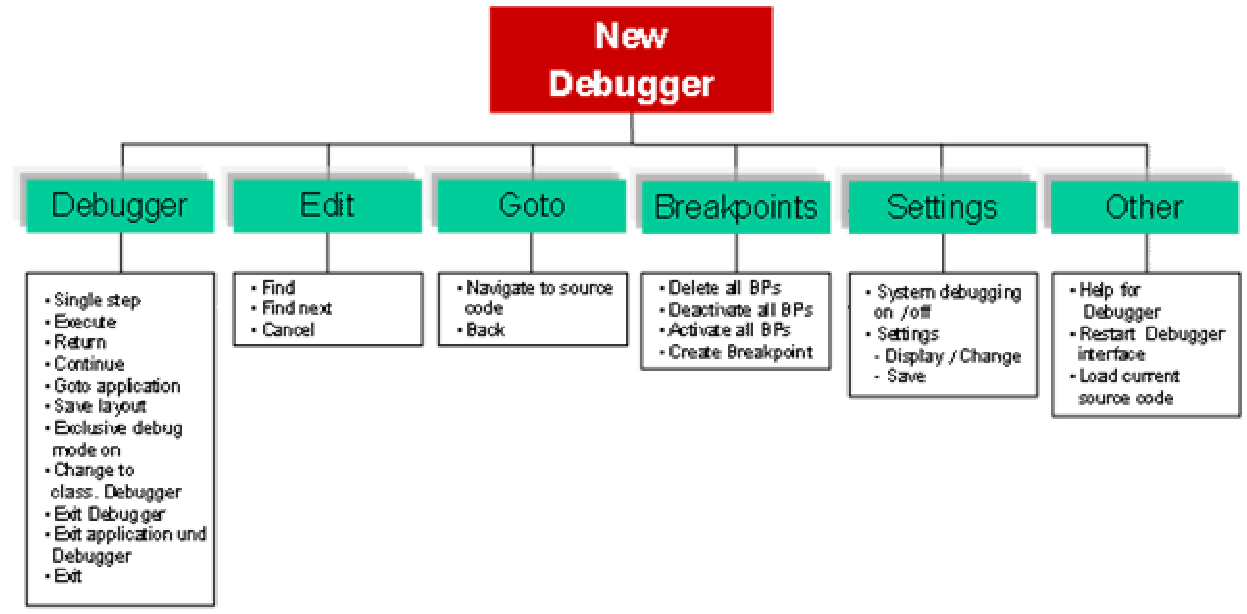
by김성준 *kimsungj@lgcns.com*



[그림 1 ABAP Debugger 종류 선택]

Classical Debugger 는 easy abap 에서 살펴 보았으므로, 이번에는 New Debugger 의 기능에 대해서 살펴보자.

New Debugger 는 기존에 비해 완전히 새롭게 디자인된 GUI 와 메뉴들로 구성되어 있으며, 이전의 디버거 기능에 새로운 기능들이 추가되었다. [그림 2]는 New Debugger 의 전체적인 모습을 설명하고 있다.

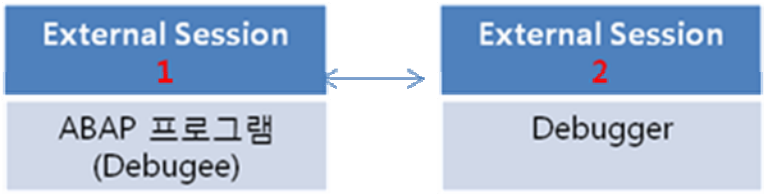


by김성준 *kimsungj@lgcns.com*

[그림 2 New Debugger 의 기능]

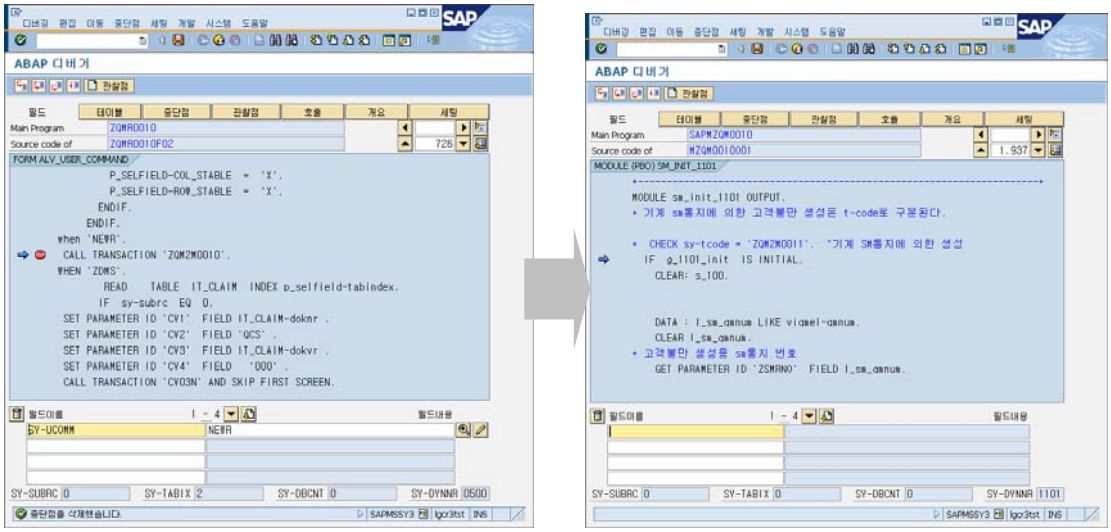
2. New debugger 와 External Session

개발자가 중단점을 설정하거나 프로그램 명령창에 /h 를 입력하면, New Debugger 가 실행된다. [그림 3]에서 ABAP 프로그램에서 디버거를 호출하면 두 개의 외부세션이 생성된다. 디버거가 활성화되면 ABAP 프로그램은 비입력 상태로 존재하다가, 디버거의 종료지점에 도달하면 ABAP 프로그램이 다시 입력 상태가 되고 디버거는 비입력 상태가 된다. 여기서 비입력 상태는 사용자가 값을 입력하거나 버튼을 클릭하는 액션을 의미한다.



[그림 3 어플리케이션과 디버거의 관계]

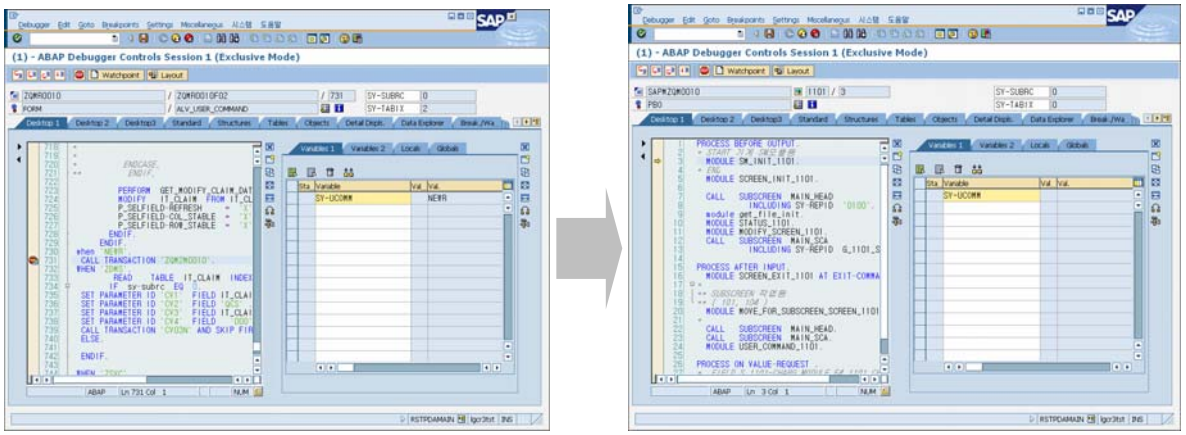
Classic Debugger 는 프로그램마다 하나의 roll area 를 사용하기 때문에 roll area 를 벗어나게 되면(SUBMIT 구문이나 CALLTRANSACTION 구문을 만나면), 디버거에서 설정했던 변수 값 조회와 같은 설정들이 모두 지워지게 된다. 예를 들어, 시스템 변수 SY-UCOMM 값을 확인하고 싶어서 [그림 4]와 같이 SY-UCOMM 을 입력했는데, CALL TRANSACTION 구문에서 다른 프로그램을 호출하게 되면 해당 설정이 사라지게 된다.



[그림 4 Classic Debugger 와 roll area 관계]

그러나 New Debugger 는 디버거는 외부세션에서 실행되기 때문에 [그림 5]와 같이 입력된 변수가 계속 남아 있게 된다.

by김성준 *kimsungj@lgcns.com*

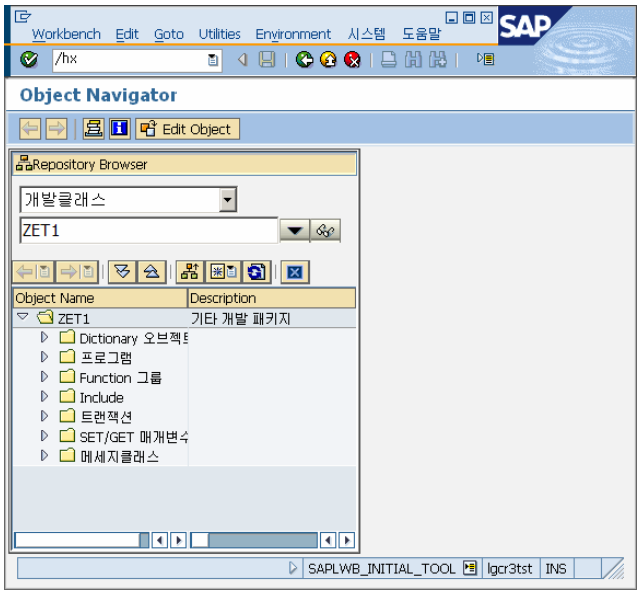


[그림 5 New Debugger 와 roll area 관계]

New Debugger 는 외부세션을 하나 더 사용하기 때문에 SAP 에 로그인한 사용자가 이미 6 개의 세션을 모두 사용하고 있다면 다음과 같은 에러 메시지가 조회되고 디버가가 실행되지 않는다.

No further external mode is available for the new ABAP Debugger

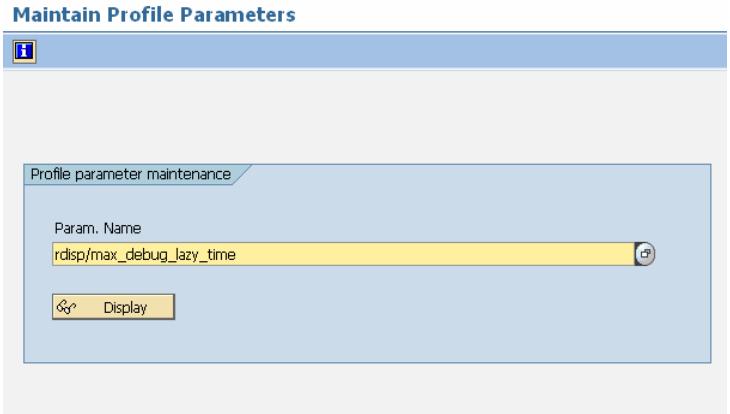
이 때에는 불필요한 화면을 하나 닫고 다시 디버깅을 호출하면 된다. 그리고 New Debugger 창이 불필요하다면, ABAP 프로그램의 명령창에 '/hx'를 입력하면 디버거 세션이 닫힌다.



[그림 6 디버거 세션 닫기]

디버거 화면은 600 초 동안 활성화되어 있다가 그 이상이 되면 자동으로 종료하게 되는데, 이것은 프로파일 파라미터 rdisp/max_debug_lazy_time 에 설정하게 된다. T-CODE: RZ11 을 실행하면 파라미터 값을 조회하거나 새로운 값을 입력할 수 있다.

by김성준 kimsungj@lgcns.com

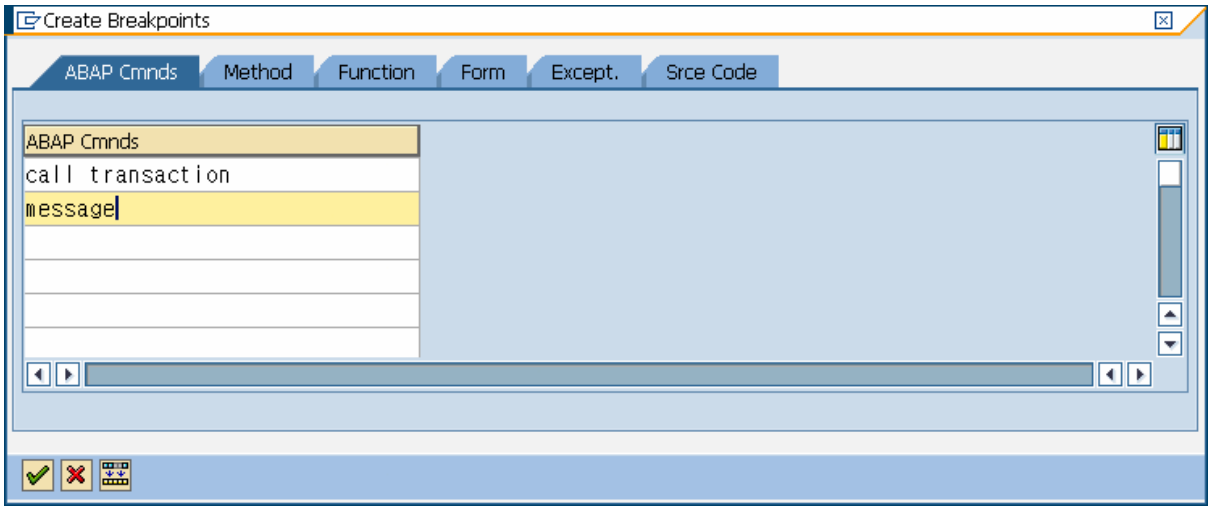


[그림 7 프로파일 파라미터 설정]

3. Breakpoint

프로그램을 디버깅 모드에서 실행하지 않고, 프로그램 내에 중단점(Breakpoint)을 설정하여 해당 구문에 도달했을 때 디버깅을 활성화할 수 있다. 어떠한 지점에서 프로그램이 Stop 한다는 의미에서 Breakpoint 라고 명명한 것이다.

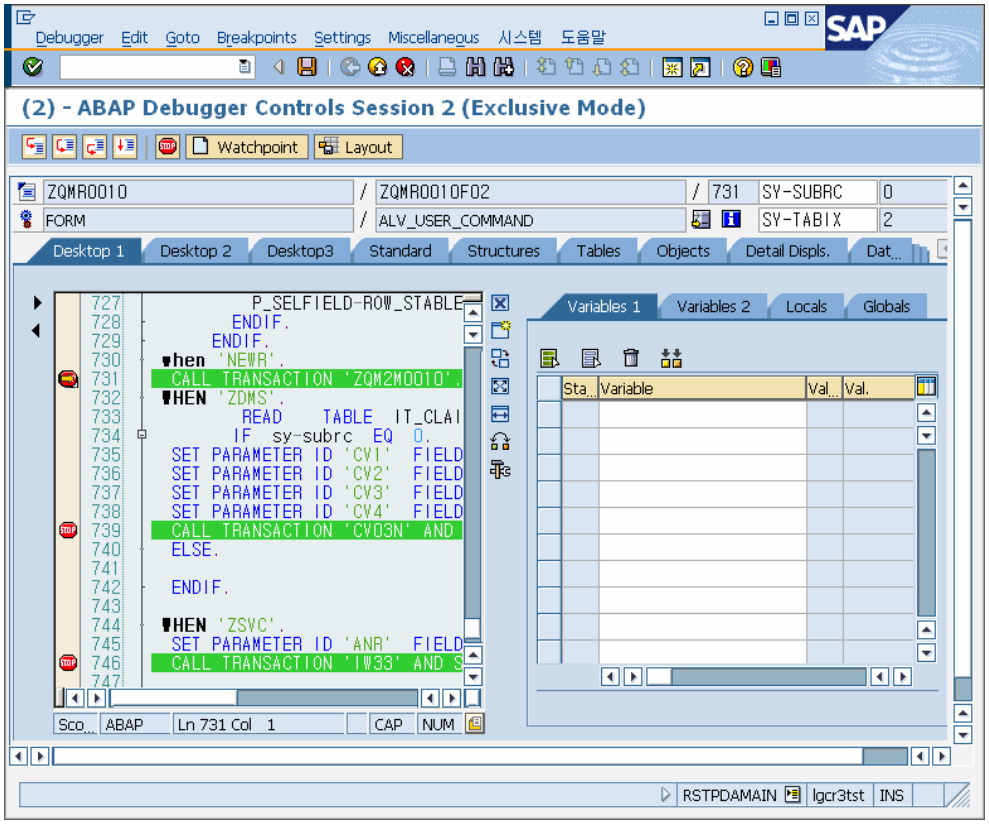
New Debugger 의 중단점은 Classic 과 거의 유사하다. New Debugger 에서는 여러 개의 중단점을 동시에 설정할 수 있다. 메뉴 : Breakpoint -> Create Breakpoint 를 선택하면, [그림 8]과 같은 화면이 열린다.



[그림 8 New Debugger 의 새로운 기능 - 중단점 동시 설정]

이 화면의 상단 탭- ABAP 명령어, 메소드, 함수, FORM 구문, 예외, 프로그램 소스-을 선택하여 중단점을 설정할 수 있다. [그림 8]은 프로그램 내의 모든 CALL TRANSACTION, MESSAGE 명령어에 중단점을 설정하게 된다. 즉, [그림 9]와 같이 중단점이 조회되는 것을 확인할 수 있다.

by김성준 *kimsungj@lgcns.com*



[그림 9 중단점이 설정된 화면]

New Debugger 에서 중단점의 종류는 [표 1] 과 같이 구분된다.(Static, Dynamic 두 부류로 분류할 수도 있다.)

아이콘	키워드	내역
	Static Breakpoint	소스코드에 직접 Break-point 명령를 입력하는 방식이다. 모든 사용자에게 활성화된다. Break username 을 입력하면 해당 사용자에게만 디버깅이 활성화 된다.
	External Breakpoint	ABAP 에디터에서 메뉴를 선택해서 설정한 중단점이다. 중단점을 설정한 사용자에게 2 시간 동안 활성화되고, SAP 에 다시 로그인해도 유지되는 중단점이다.
	Session Breakpoint	ABAP 에디터에서 메뉴를 선택해서 설정한 중단점이다. 즉, 외부세션에 설정된 중단점 SAP 에 로그인한 사용자가 열수 있는 6 개의 창에 모두 적용된다.
	Dynamic Breakpoint	디버거 인스턴스 활성화시에만 효력이 유지된다.

[표 1 중단점의 종류]

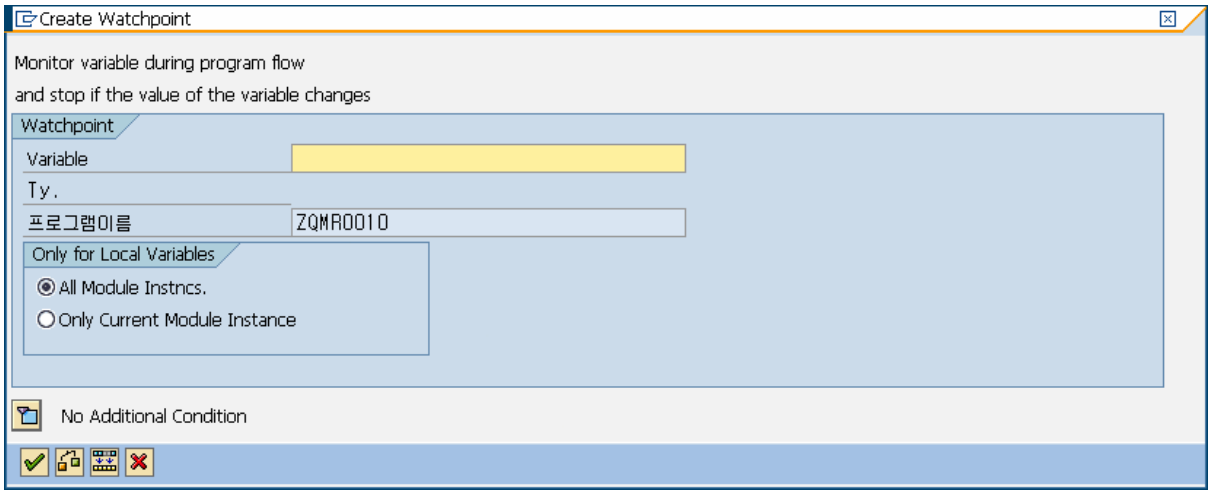
by김성준 kimsungj@lgcns.com

4.Watchpoint

프로그램을 개발하면서 변수가 원하는 값이 아닌 예외적인 값들로 조회되는 경우를 자주 접하게 된다. 또는 분명히 CLEAR 구문으로 값을 초기화하였는데도 값이 지워지지 않는 경험도 종종 하게 된다. 도대체 어느 부분에서 변수 값이 변경되는지 알 수 없을 때 Watchpoint 를 사용하면 아주 유용하게 원인을 찾아낼 수 있다. 그리고 다른 개발자가 개발하였거나 표준 프로그램에서 변수 값이 어떠한 절차로 변경되는지 추적할 때도 아주 큰 도움이 된다. 즉, Watchpoint(관찰점)는 프로그램 실행 도중 조건 값으로 변경될 때 프로그램을 정지시키는 기능을 제공한다. 다만, 관찰점은 프로그램 내의 구조, Internal Table, 필드의 값이 변경된 시점에 활성화된다는 점에서 중단점과 큰 차이가 있다.

New Debugger 화면의 툴바에서 

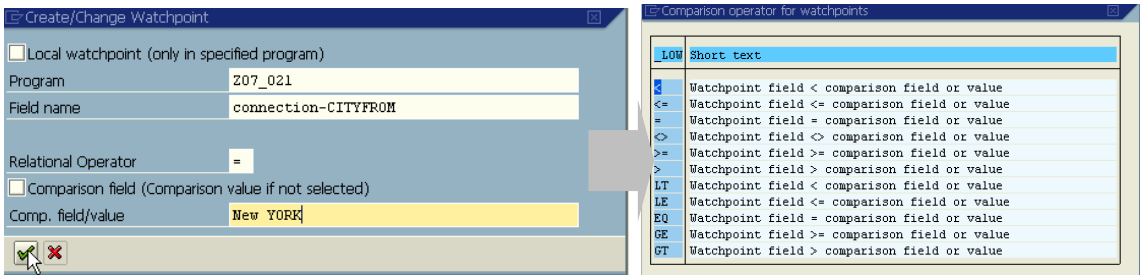
버튼을 선택하면 관찰점을 생성할 수 있다.



[그림 10 관찰점 생성 화면]

Variable 필드에 변수를 입력하고 확인버튼을 클릭하면 관찰점이 생성된다. 프로그램 수행시 해당 변수의 값이 변경되면 ***Watchpoint reached (변수명)*** 라는 메시지와 함께 정지되고 디버거가 활성화된다.

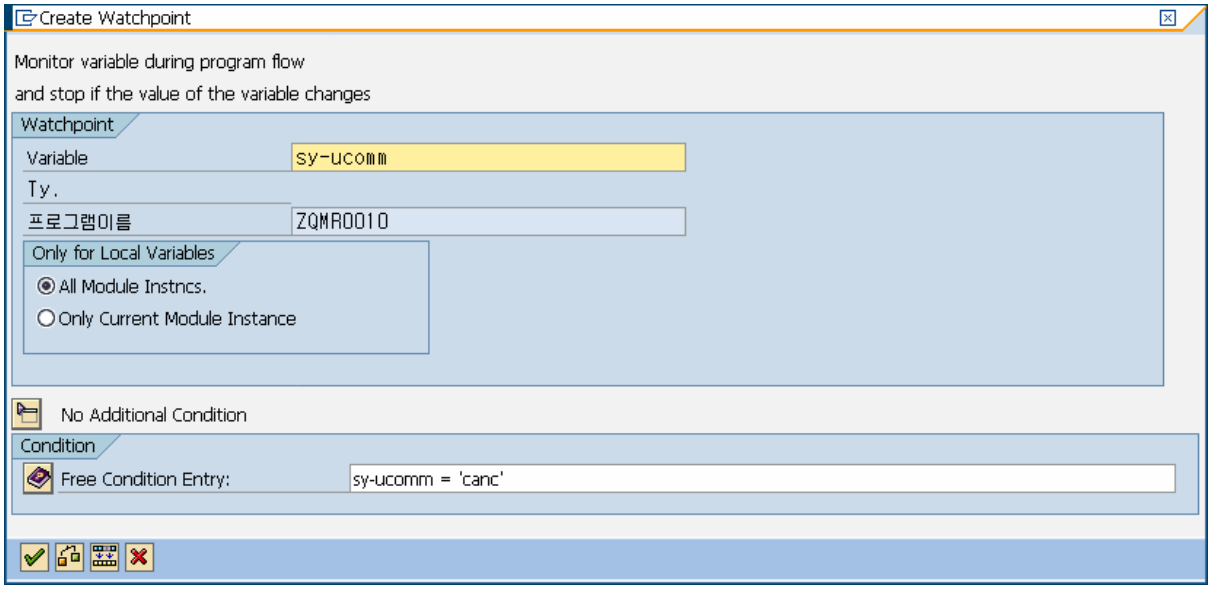
[그림 11]과 같이 Classic Debugger 에서는 변수의 값을 관계연산자를 이용하여 값을 비교하여 조건을 설정할 수 있었다.



[그림 11 Classic Debugger 에서의 관계연산자 입력화면]

by김성준 *kimsungj@lgcns.com*

New Debugger 에서는 관계 연산자 뿐만 아니라 명령어를 추가하여 조건에 추가할 수 있다. [그림 12]는 sy-ucomm 이라는 시스템 변수가 'canc' 라는 명령어가 지정될 때 관찰점을 활성화하라는 뜻이다.



[그림 12 New debugger 에서의 조건추가]

 버튼을 선택하면 열리는 Free Condition Entry(관찰점 추가 조건)를 입력할 수 있다. 여기에 사용할 수 있는 조건의 문장은 다음의 규칙을 따라야 한다.

<Function(Variable) or Variable> Operator <Function(Variable) or Variable or Constant>

예를 들면, string 변수 l_1 과 l_2 가 있다고 하면 두 변수의 길이가 같아 질 때 관찰점을 설정할 경우 다음과 같은 조건을 주면 된다.

strlen(l_1) = strlen(l_2)

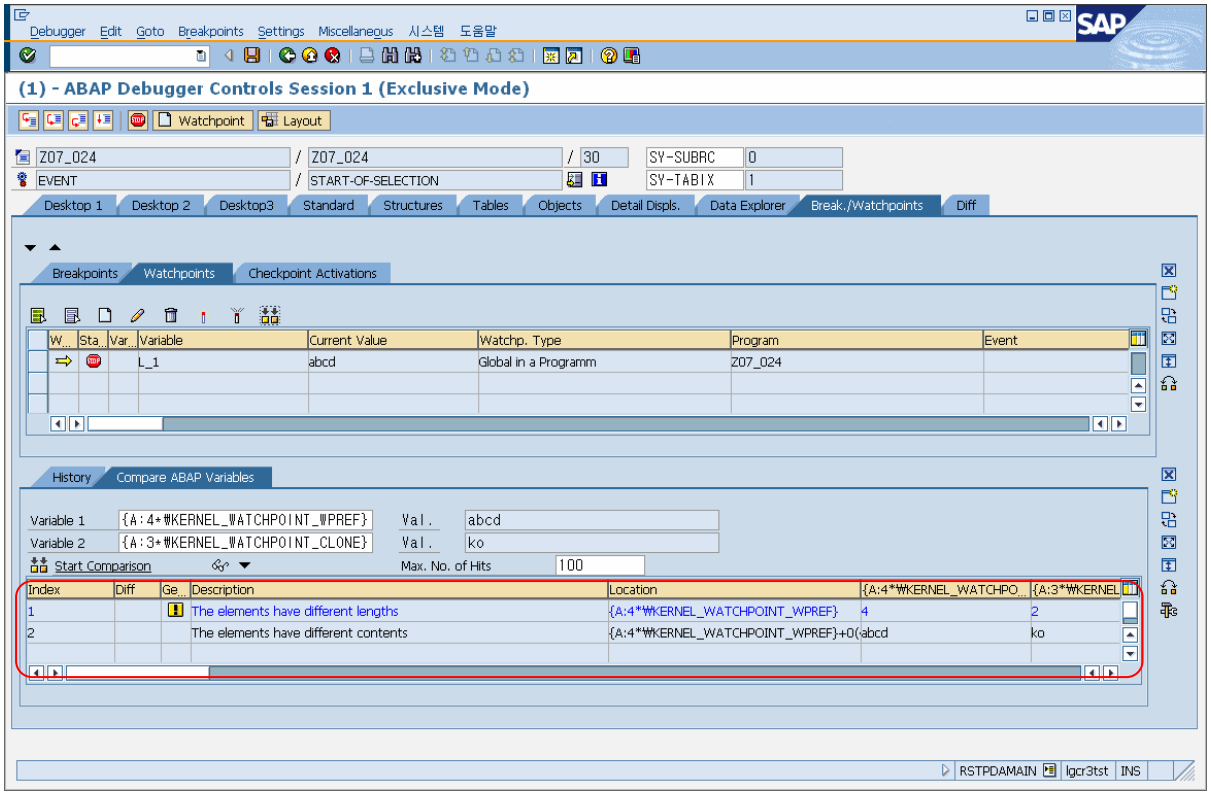
또 다른 예를 들면, 인터널 테이블 itab 의 5 번째 라인이 생성될 때 관찰점을 활성화하고 싶은 경우는 다음과 같이 추가할 수 있다.

Lines(itab) = 5

 버튼은 관찰점을 추가하고, 다시 화면을 초기화하여 새로운 관찰점을 생성할 수 있도록 한다.

관찰점이 추가되어 프로그램이 실행되는 도중 관찰점 지점에 도착하면 Break/Watchpoint 탭 -> Watchpoint 탭에서  버튼을 선택하면 변수의 변경 이전/이후의 값을 비교할 수 있다.

by김성준 *kimsungj@lgcns.com*



그리고 다음 버튼들을 이용해 관찰점 및 중단점을 생성, 변경, 삭제, 활성화, 비활성화를 할 수 있다.

이외의 디버거 화면들은 GUI 를 통해 직관적으로 알 수 있을 것이라 생각하고 여기서 마무리한다.

참고 :
http://help.sap.com/saphelp_nw70ehp1/helpdata/en/84/1f624f4505144199e3d570cf7a9225/frameset.htm

by김성준 *kimsungj@lgcns.com*