7. Action Block

Dieses Kapitel beschäftigt sich näher mit der Programmlogik, bzw Action Blocks.
In einem Action Block findet sich der „Quellcode“ eines Dialogs. Theoretisch kann man einen Action Block mit einer Funktion vergleichen. Sie hat Eingabeparameter (Import), Rückgabewerte (Export) und einen Programmrumpf.

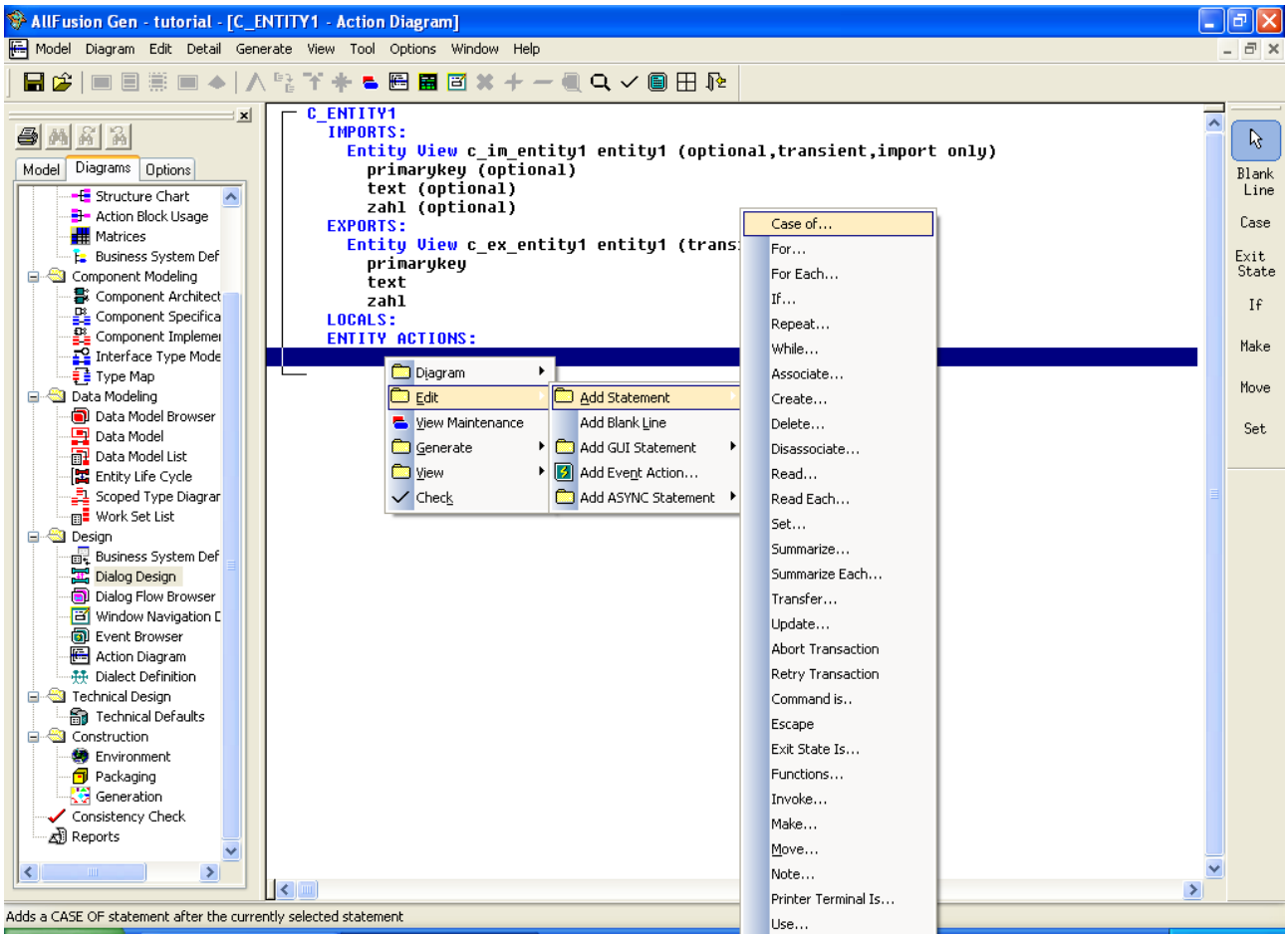
Action Blocks werden aufgerufen wenn z.B. ein bestimmtes Ereignis auf der Oberfläche eintritt. Ein Beispiel hierfür ist das Drücken eines Buttons. Wenn ein Button als „Button Action“ die Option „Execute Procedure Step“ ausgewählt hat, wird der Action Block gestartet sobald man den Button drückt.

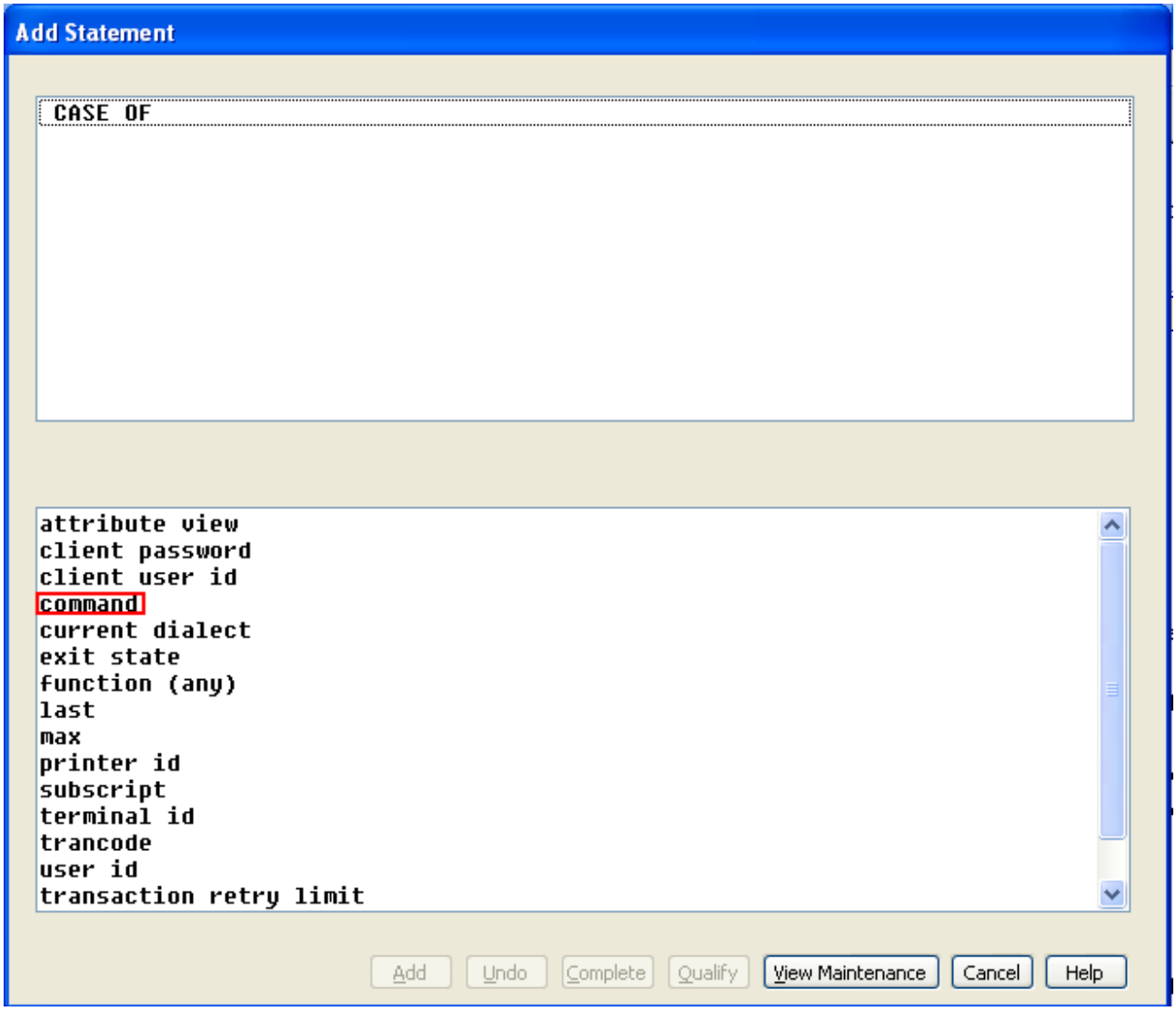
Öffnen Sie zunächst wieder den Action Block des Clients ihrer Entität (hier C_Entity1) über das Dialog Design.

Unter der Zeile „Entity Actions“ befindet sich der eigentliche „Quellcode“. Bisher steht hier jedoch nichts.

Tipp: Mit STRG + B oder Rechtsklick – Edit – Add Blank Line können Sie leere Zeilen hinzufügen.

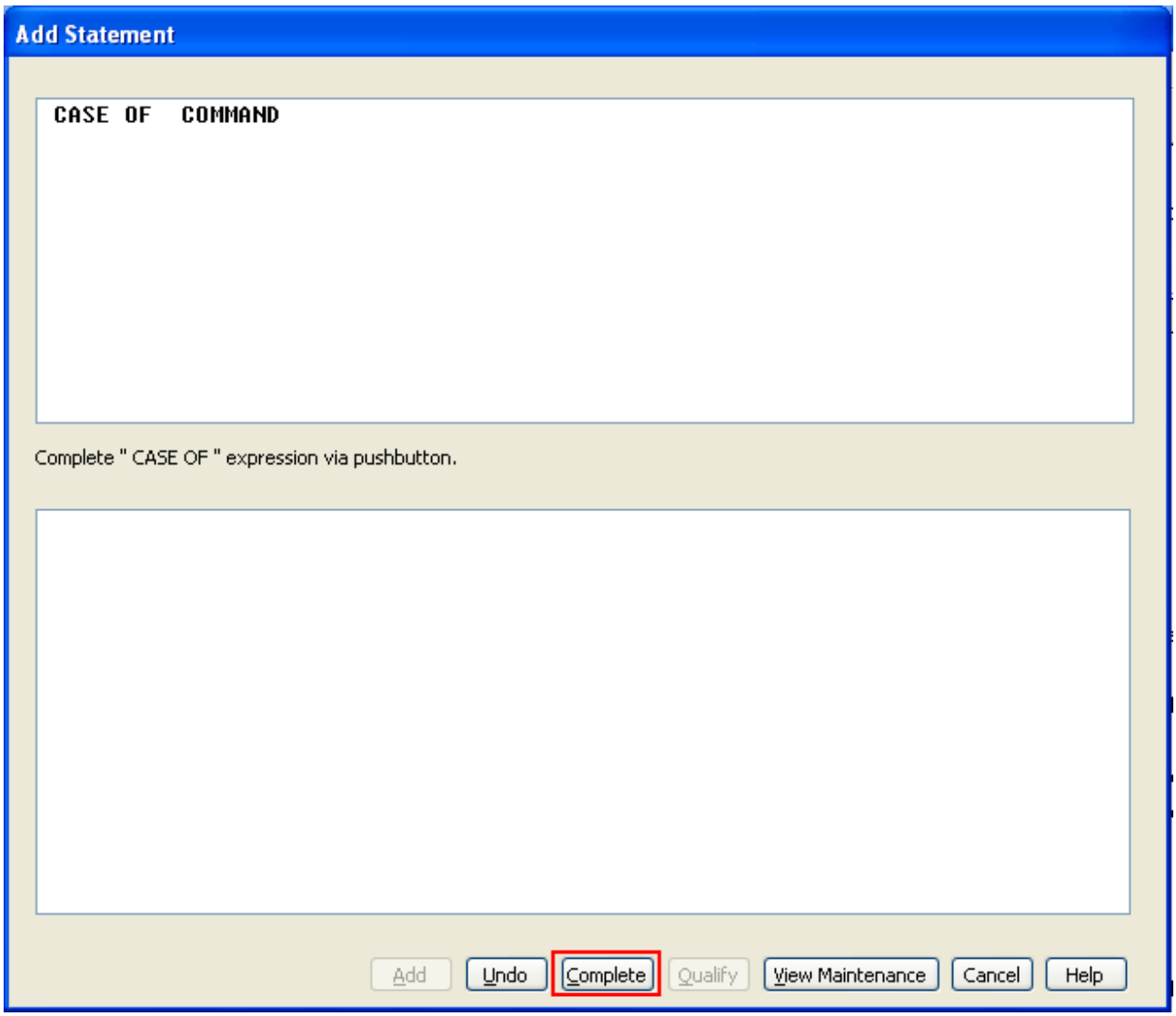
Machen Sie einen Rechtsklick auf die leere Zeile unter Entity Actions. Unter Edit – Add Statement finden sie alle Befehle, die sie hier einfügen können. Wählen Sie das Statement „Case of...“. Wie auch in anderen Programmiersprachen dient „Case of...“ zur Fallunterscheidung. Sie können damit zum Beispiel unterscheiden, welcher der Buttons auf der Oberfläche gedrückt wurde.



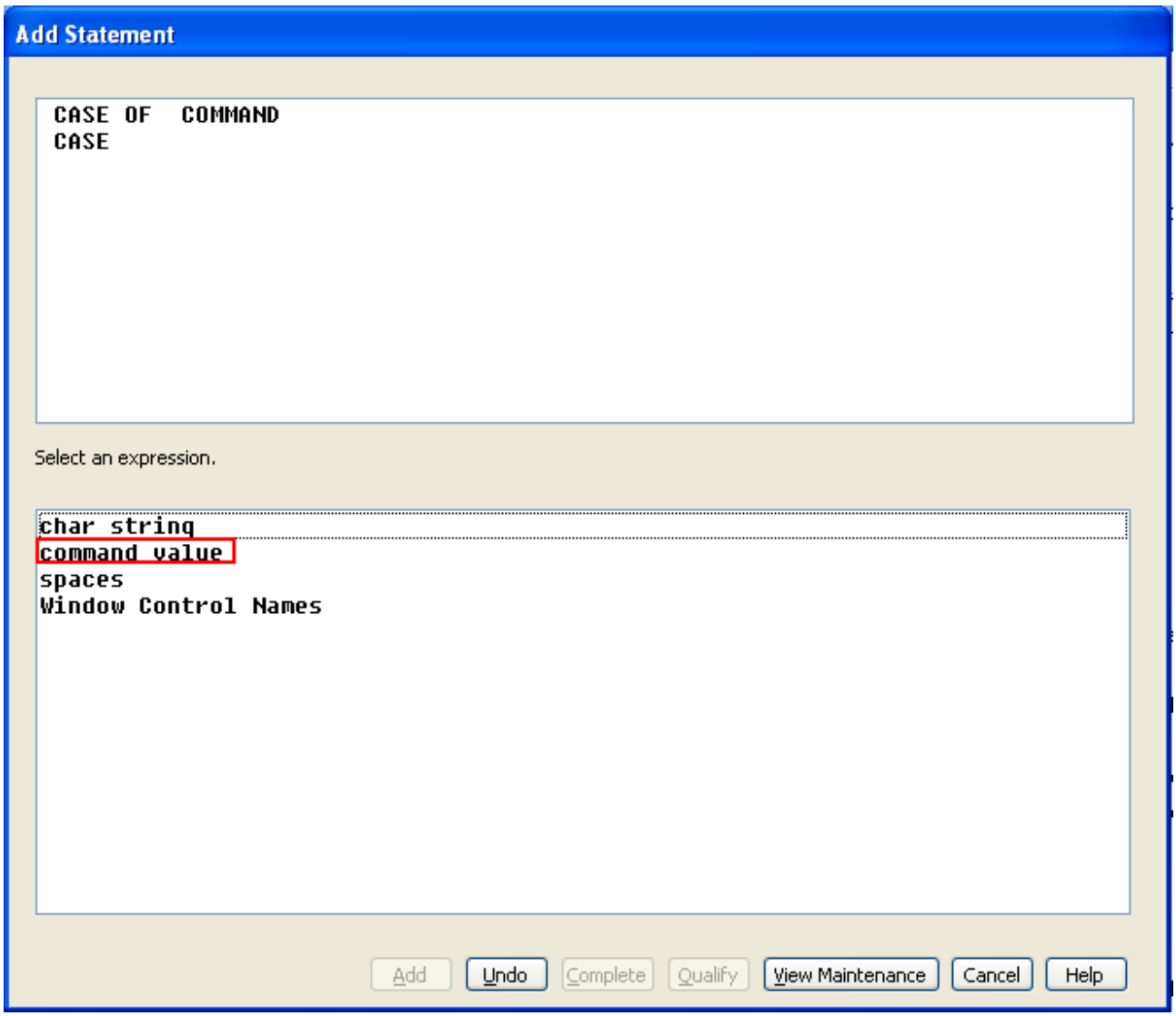


Nun wird sich ein Fenster öffnen, in dem Sie das „Case of...“ näher definieren können. Zuerst muss ausgewählt werden welchen Wert „Case of...“ untersuchen soll. Wählen Sie im Auswahlbildschirm den Eintrag „command“.

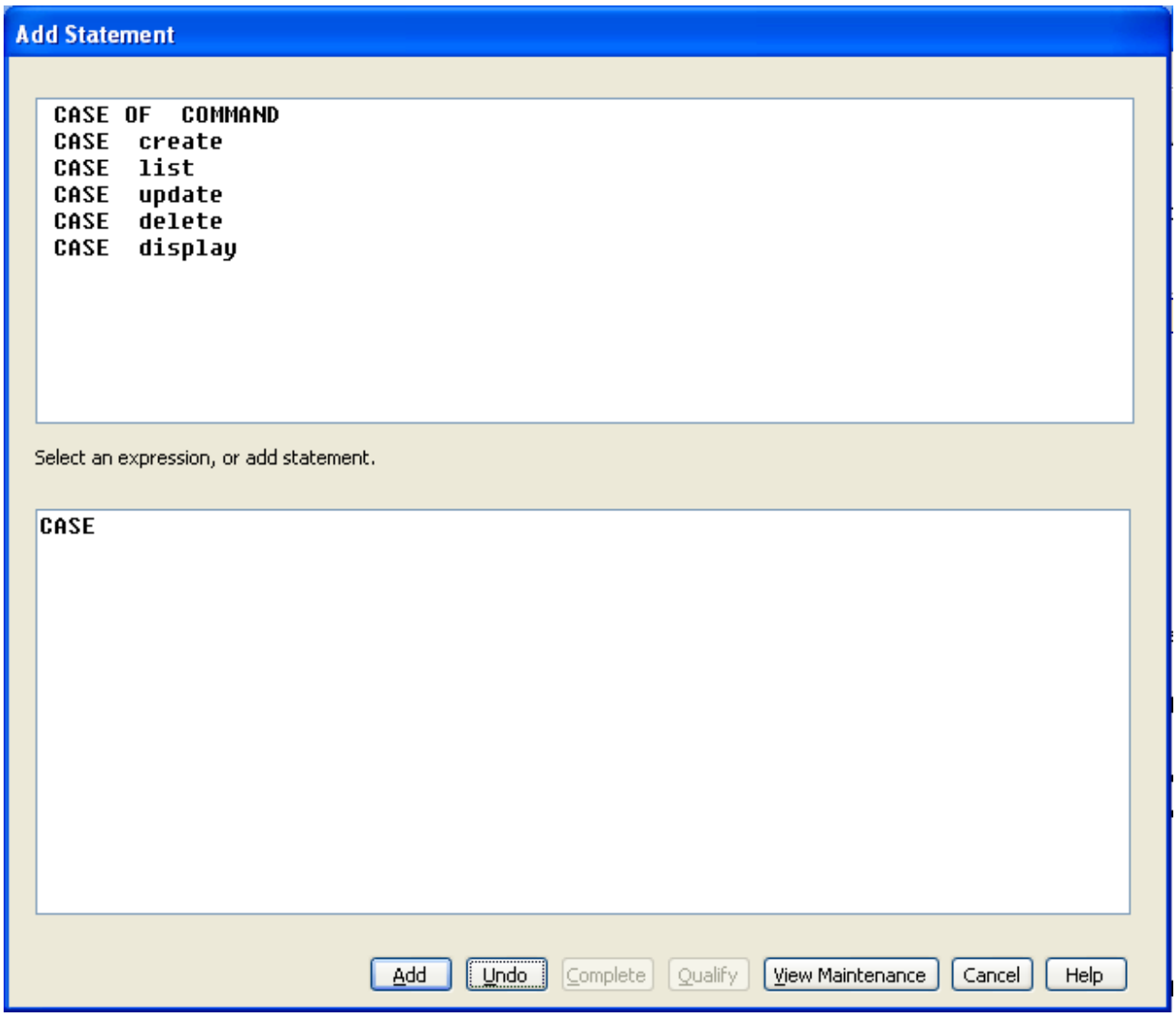
„Command“ ist eine globale Systemvariable, die gesetzt und ausgelesen werden kann. Das setzten übernehmen die Buttons auf der Oberfläche, wenn Sie jedem Button einen Befehl zugeordnet haben. Das auslesen übernimmt nun das „Case of...“.



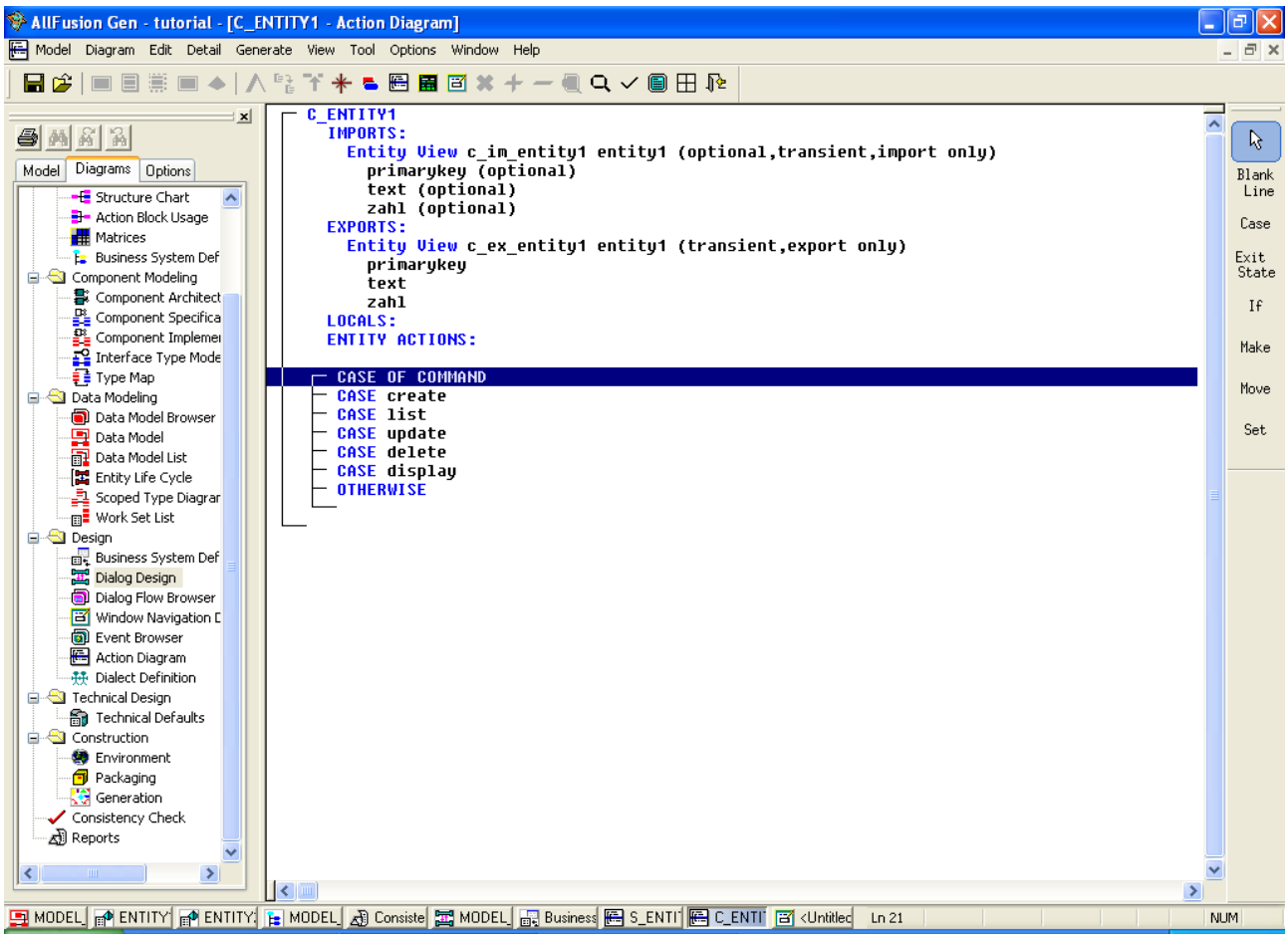
Nachdem Sie auf „command“ geklickt haben, drücken Sie anschließend auf „Complete“. Dadurch ist es möglich die einzelnen Fälle schon in diesem Fenster zu definieren. Alternativ kann man auch im Action Block ein Statement „Case“ hinzufügen, wenn man beispielsweise einen der Fälle vergessen hat und ihn nachtragen möchte.



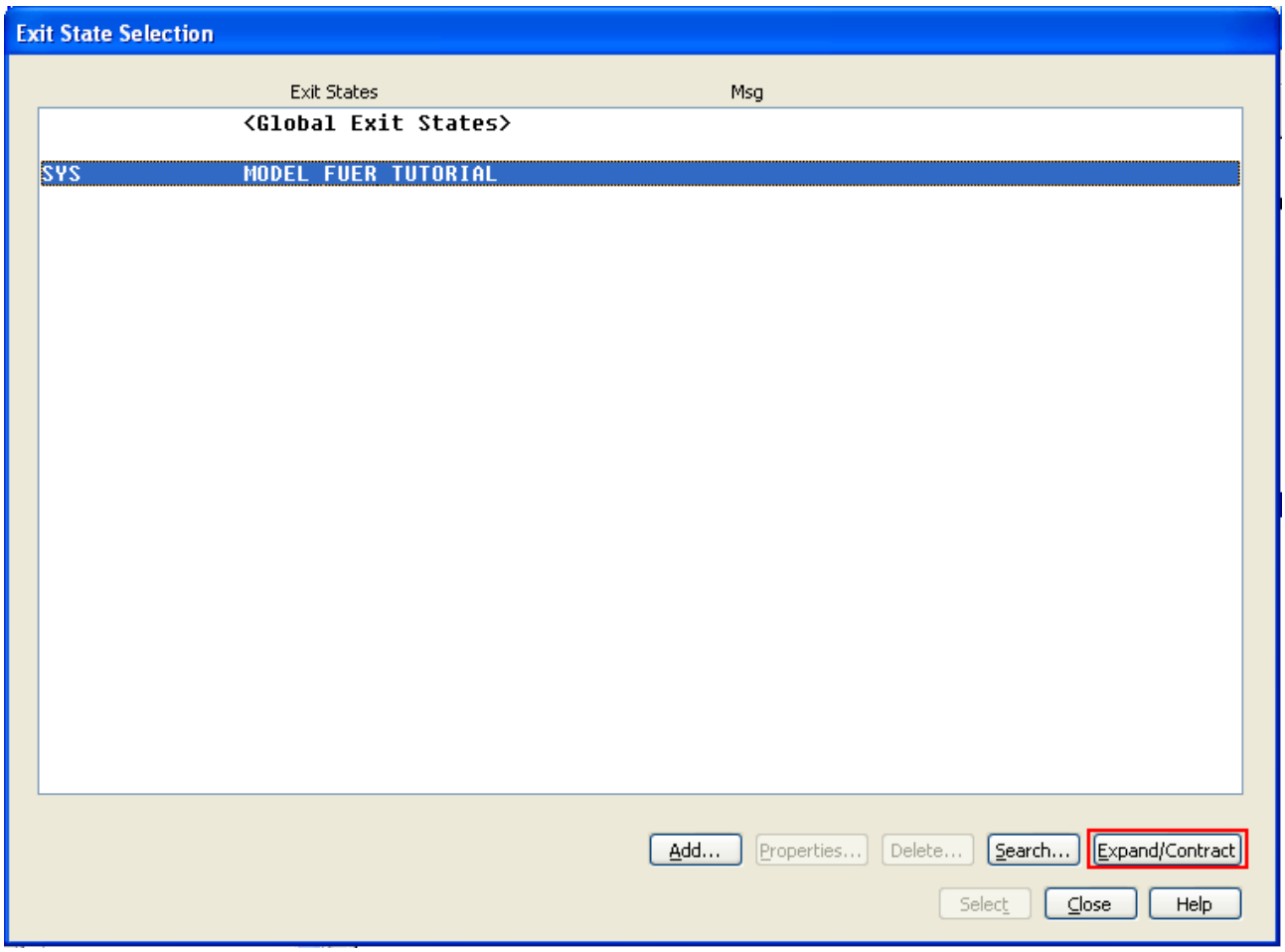
In der oberen Hälfte des Bildschirms können Sie erkennen wie Ihr Statement bisher aussieht. Der unterste Eintrag ist zur Zeit „Case“. Es wird nun ein konkreter Wert erwartet. Wählen Sie unten den Eintrag „command value“ und als nächstes einen der Befehle, die Sie für die Buttons ausgesucht haben.



Anschließend klicken Sie unten wieder auf „Case“ um einen weiteren Fall hinzuzufügen. Wenn Sie alle Ihre Befehle so eingefügt haben, klicken Sie auf „Add“ um das zusammengebaute Statement einzufügen.



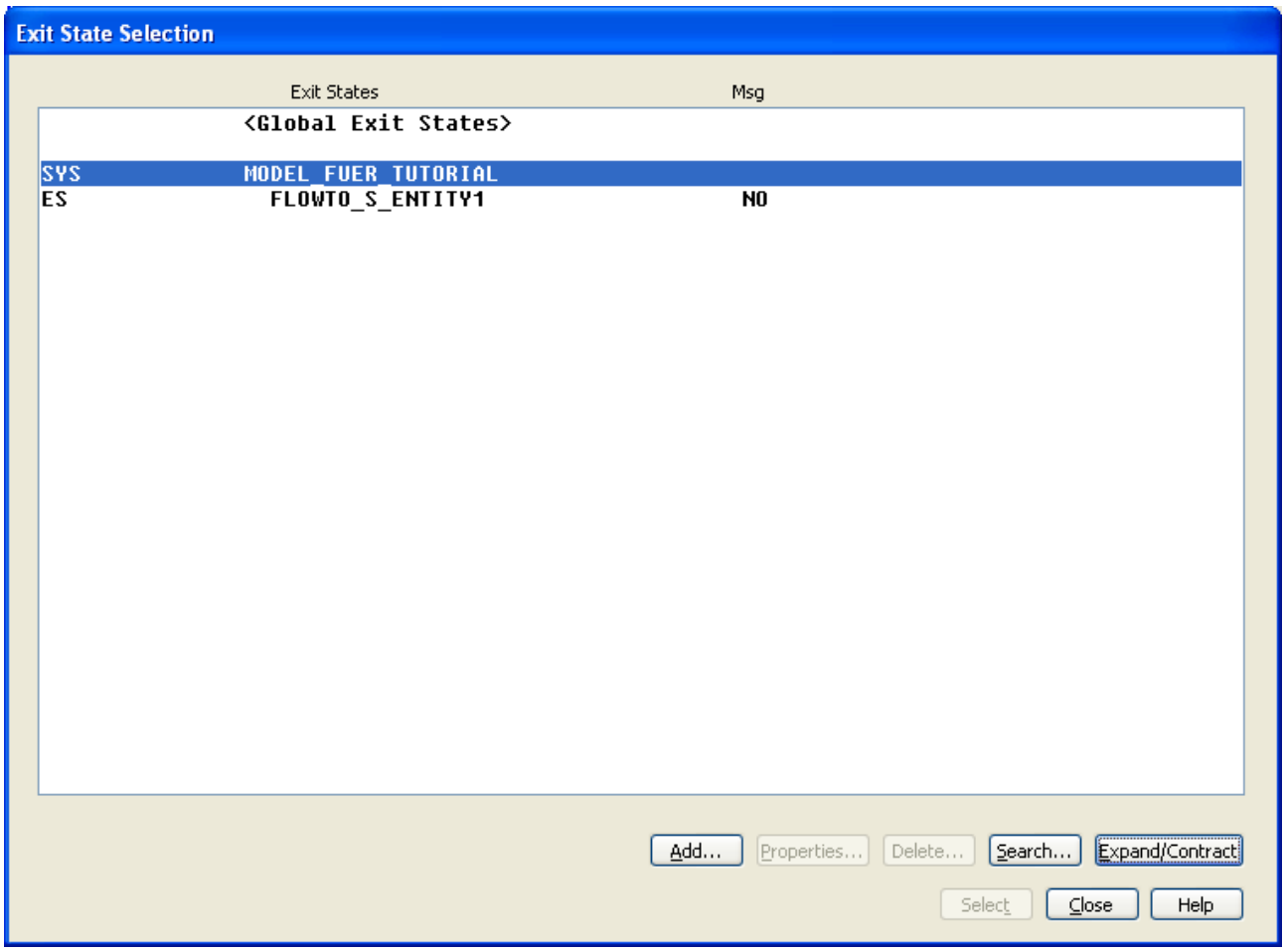
Wenn alles geklappt hat, sieht Ihr Action Block in etwa wie in diesem Bild aus. Als nächstes fügen Sie zu jedem der „Case“ Statements ein „Exit State“ ein. Der Befehl ist sowohl in Edit – Add Statement als auch auf der Rechten Seite des Fensters als Shortcut zu finden.



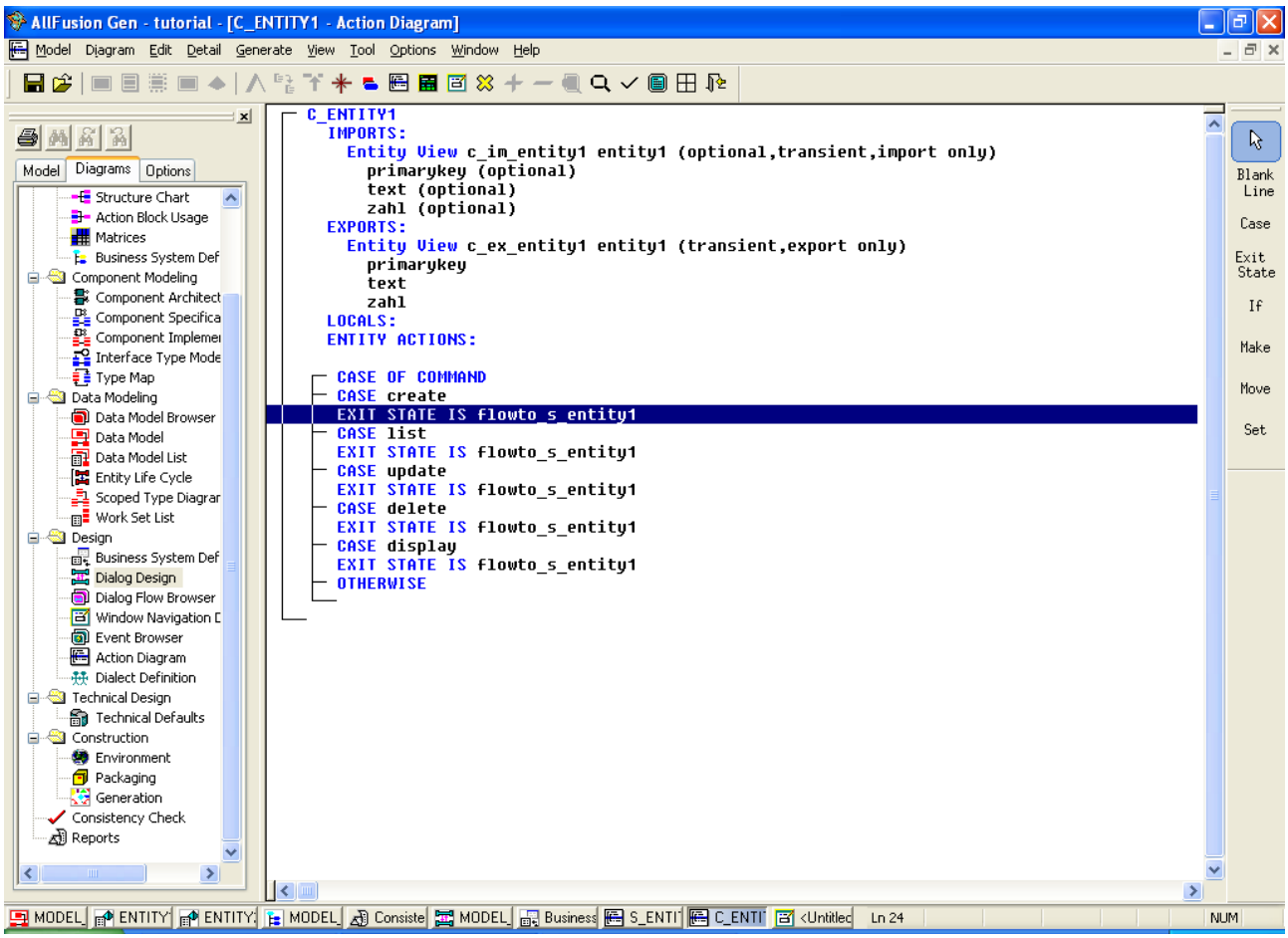
Hier haben sie nun die Übersicht über alle Exit States. Ein Exit State ist wie „Command“ eine globale Systemvariable. Mit seiner Hilfe können Sie den Datenaustausch zwischen Client- und Serverdialog realisieren.

Zunächst wählen Sie ihr aktuelles Business System aus und klicken auf „Add...“.

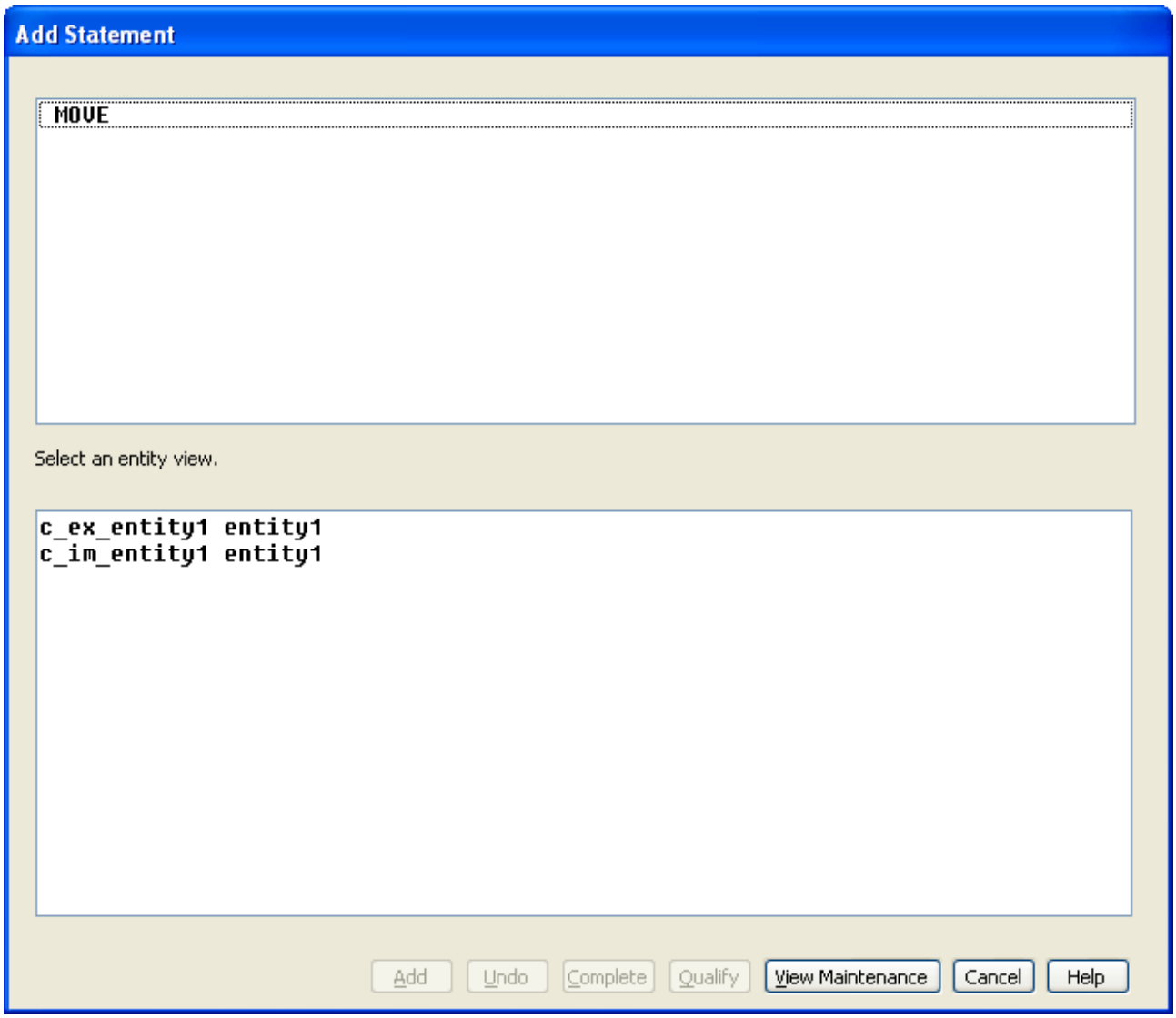
Geben Sie dem Exit State einen Namen. Bedenken Sie, dass sie später für jede Entität mindestens ein solchen Exit State brauchen. Diese müssen aber noch nicht jetzt angelegt werden, sondern können auch später erzeugt werden, damit Sie sie gleich zu den richtigen Business Systemen zuordnen können.
Lassen sie Message Type auf „None“.



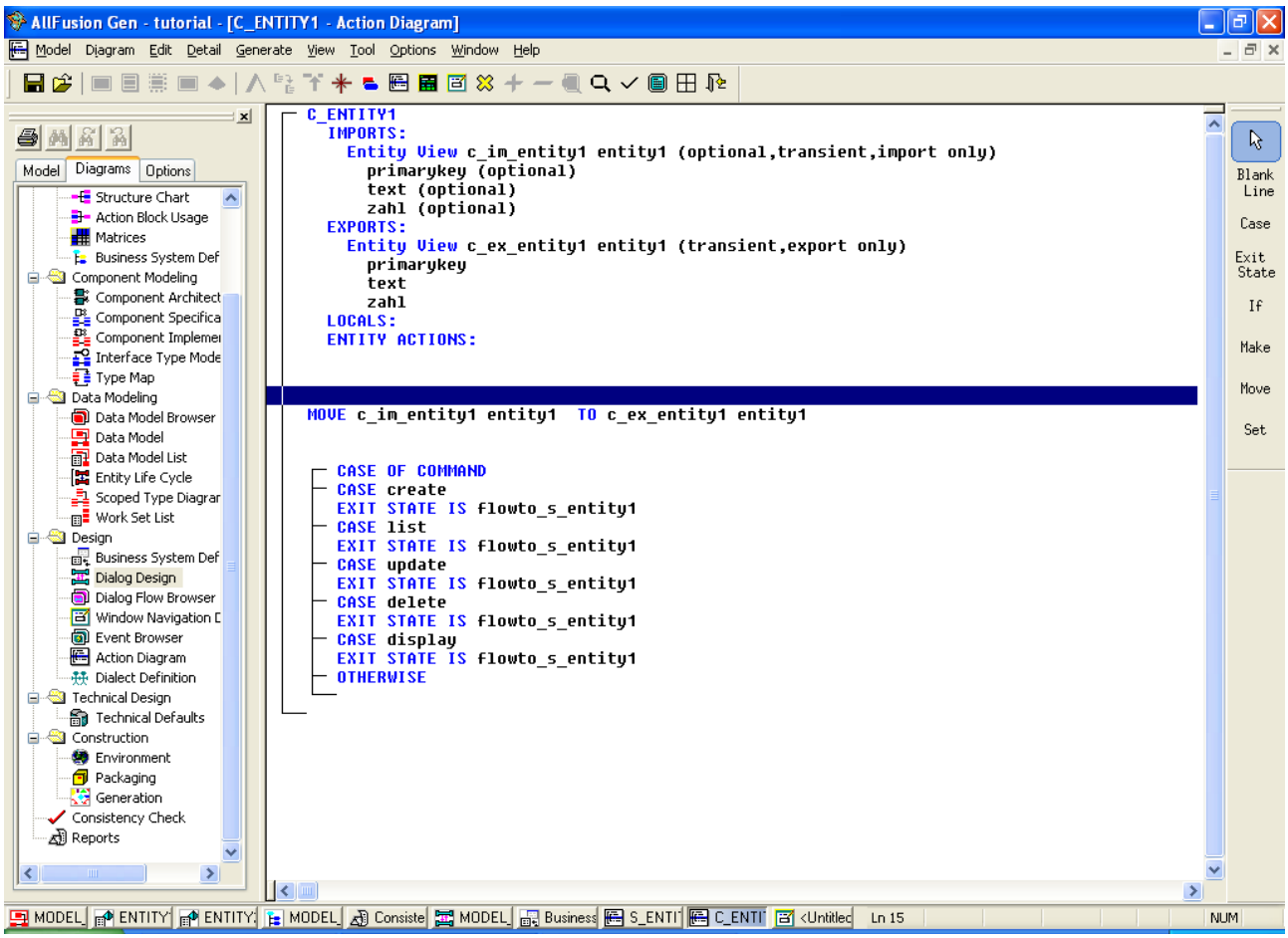
Wenn Sie nun das Business System auswählen und unten rechts auf „Expand/Contract“ klicken, sehen Sie Ihren Exit State. Die Zeile „Global Exit States“ kann auch ausgefahren werden. Dort sehen Sie alle vom System bereitgestellten Exit States.



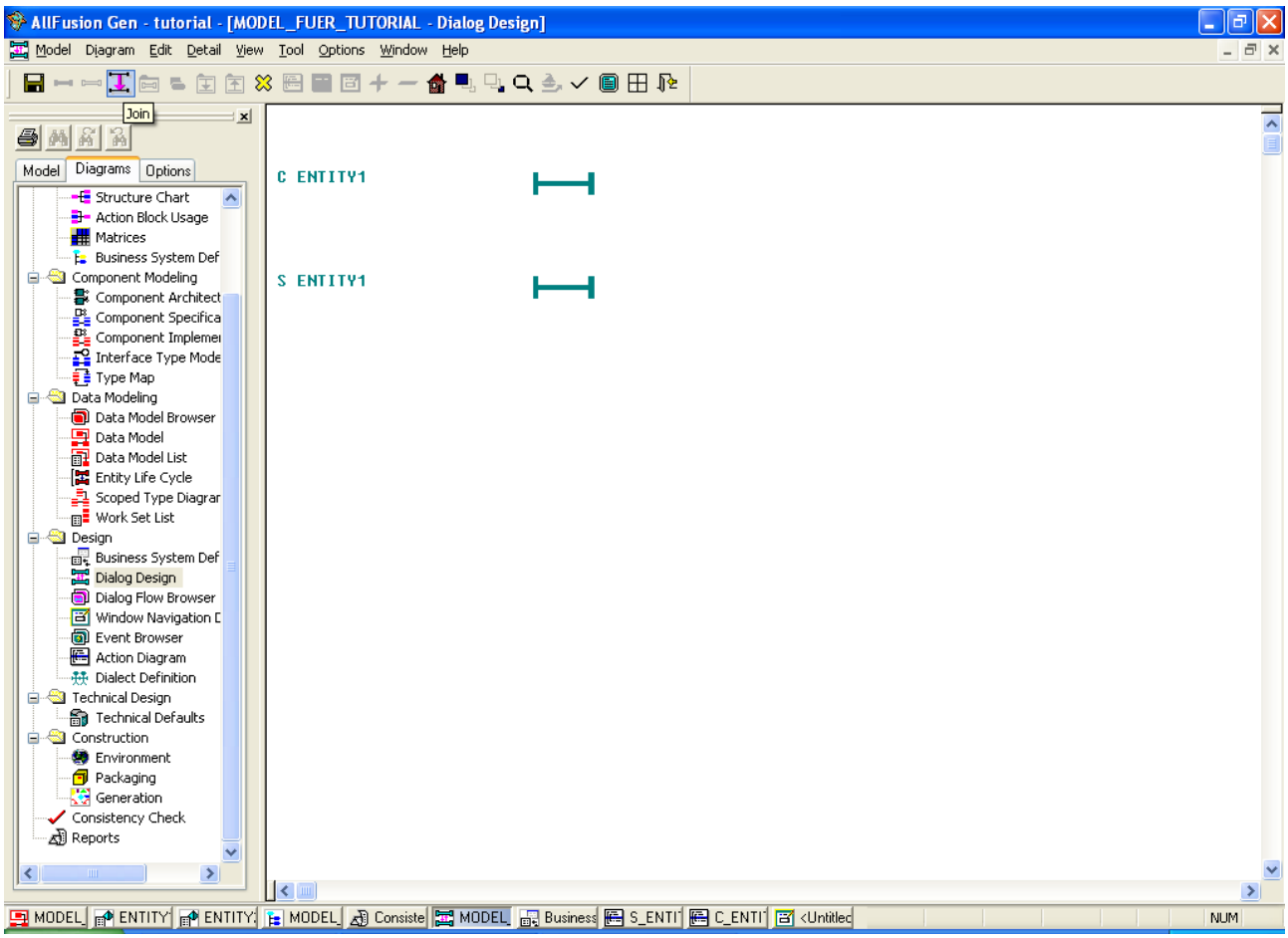
Nun sollen die Daten des Importviews in den Exportview kopiert werden, damit sie an den Server weitergereicht werden können. Hierzu wählen Sie die leere Zeile über dem „Case of“ an und fügen ein „Move“ Statement hinzu.



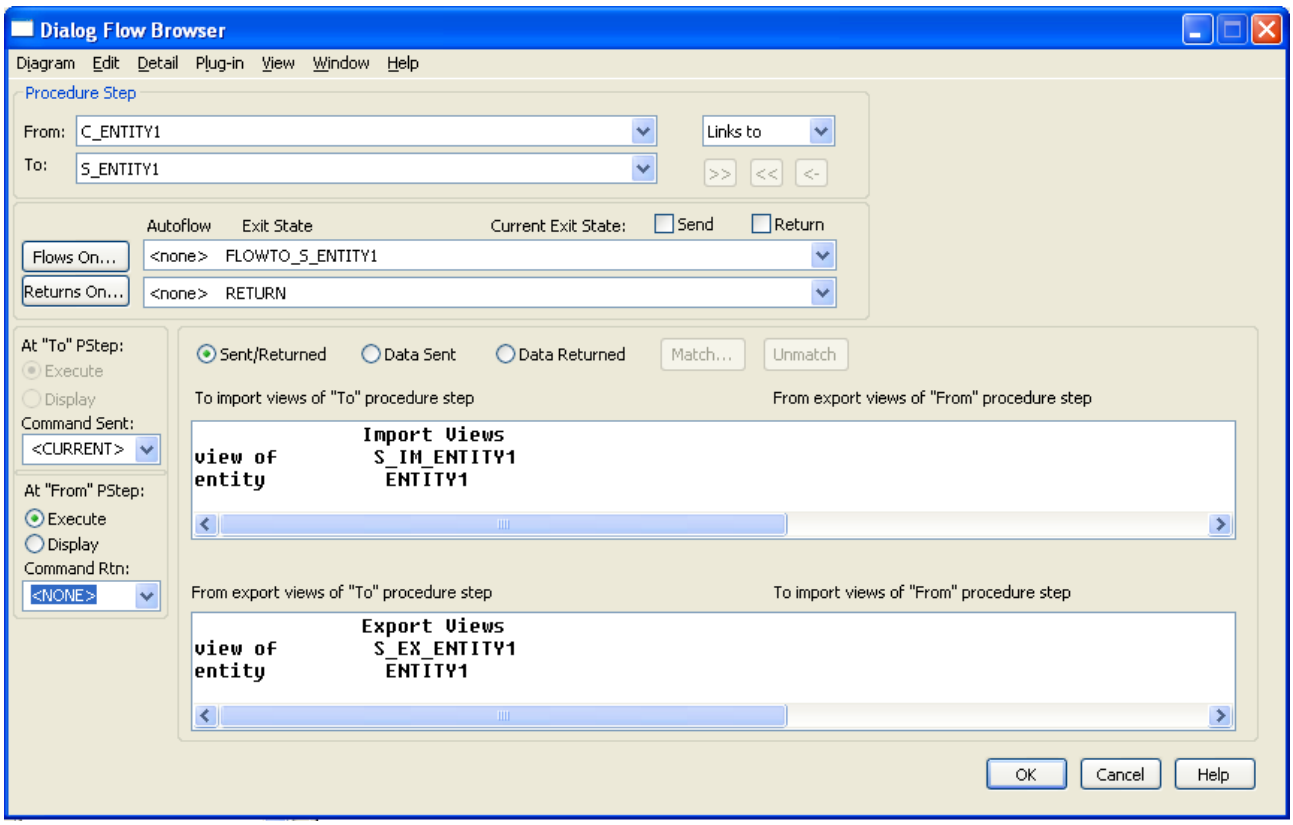
Move ist ein Befehl welcher den Inhalt einer Entityview in eine andere Entityview kopiert. Wie Sie sehen haben Sie zwei Views zur Auswahl (die beiden einzigen Entityviews in diesem Action Block). Wählen Sie die Importview aus und klicken sie danach auf Add. Die Exportview sollte automatisch als Ziel ausgewählt werden, da sie die einzige gültige Wahl ist.



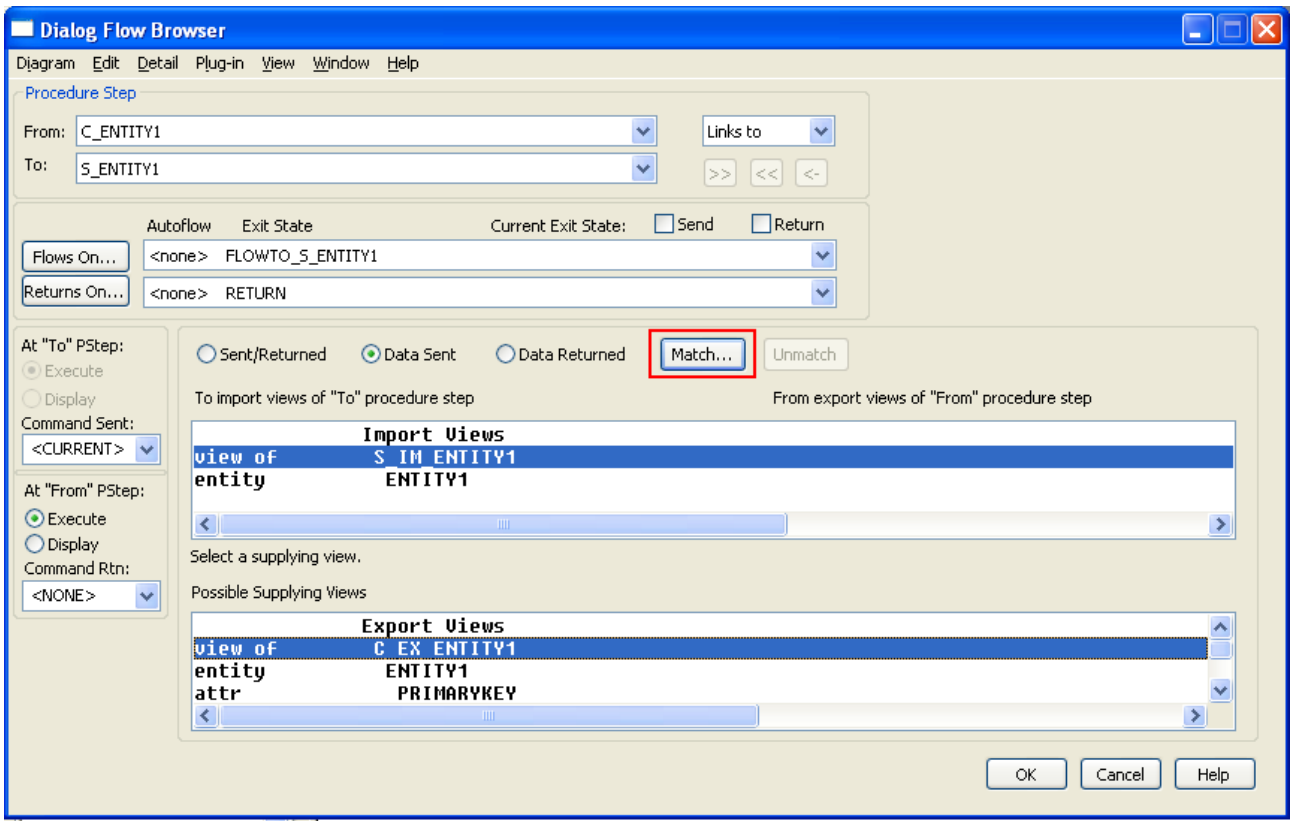
Damit ist vorerst der Ablauf des Clients fertig.



Nun soll der Datenaustausch zwischen Client und Server implementiert werden. Wechseln Sie dazu wieder ins Dialog Design. Wählen Sie (mit gedrückter STRG Taste) beide Dialoge aus. Achten Sie darauf, dass der Client zuerst angewählt wird. Anschließend drücken Sie auf „Join“ (oben links).

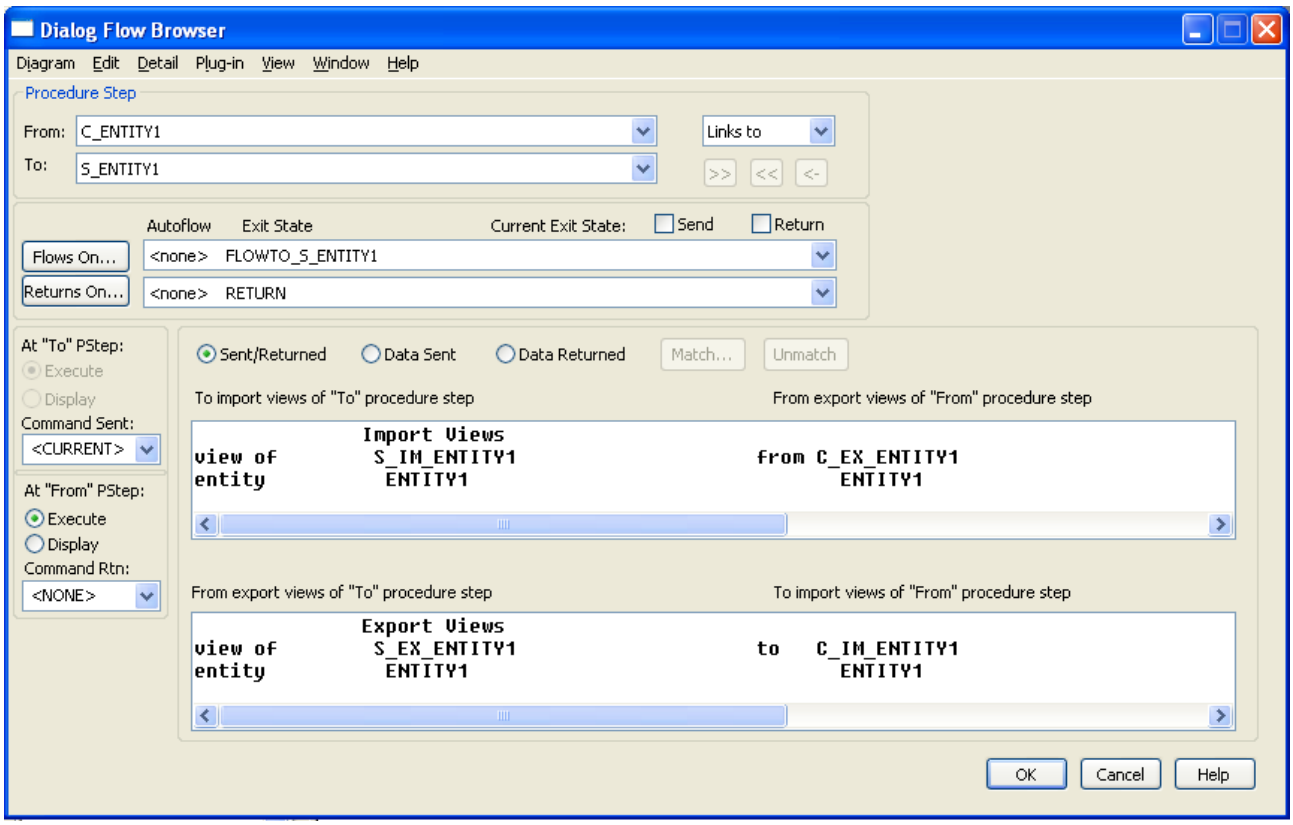


Das sich nun öffnende Fenster dient zur Einstellung des neuen Joins, bzw dem Datenfluss. „From“ ist der ausgehende Dialog, „To“ das Ziel. Achten Sie darauf, dass die Reihenfolge hier richtig ist. Klicken Sie auf „Flows On“ und wählen Sie den von Ihnen erzeugten Exit State aus. Bei „Returns On“ können Sie den globalen Exit State „Return“ nutzen.
Als „Command Sent“ wählen Sie „Current“ aus, damit der aktuelle Befehl weitergereicht wird. „Command Rtn“ setzen Sie auf „none“.

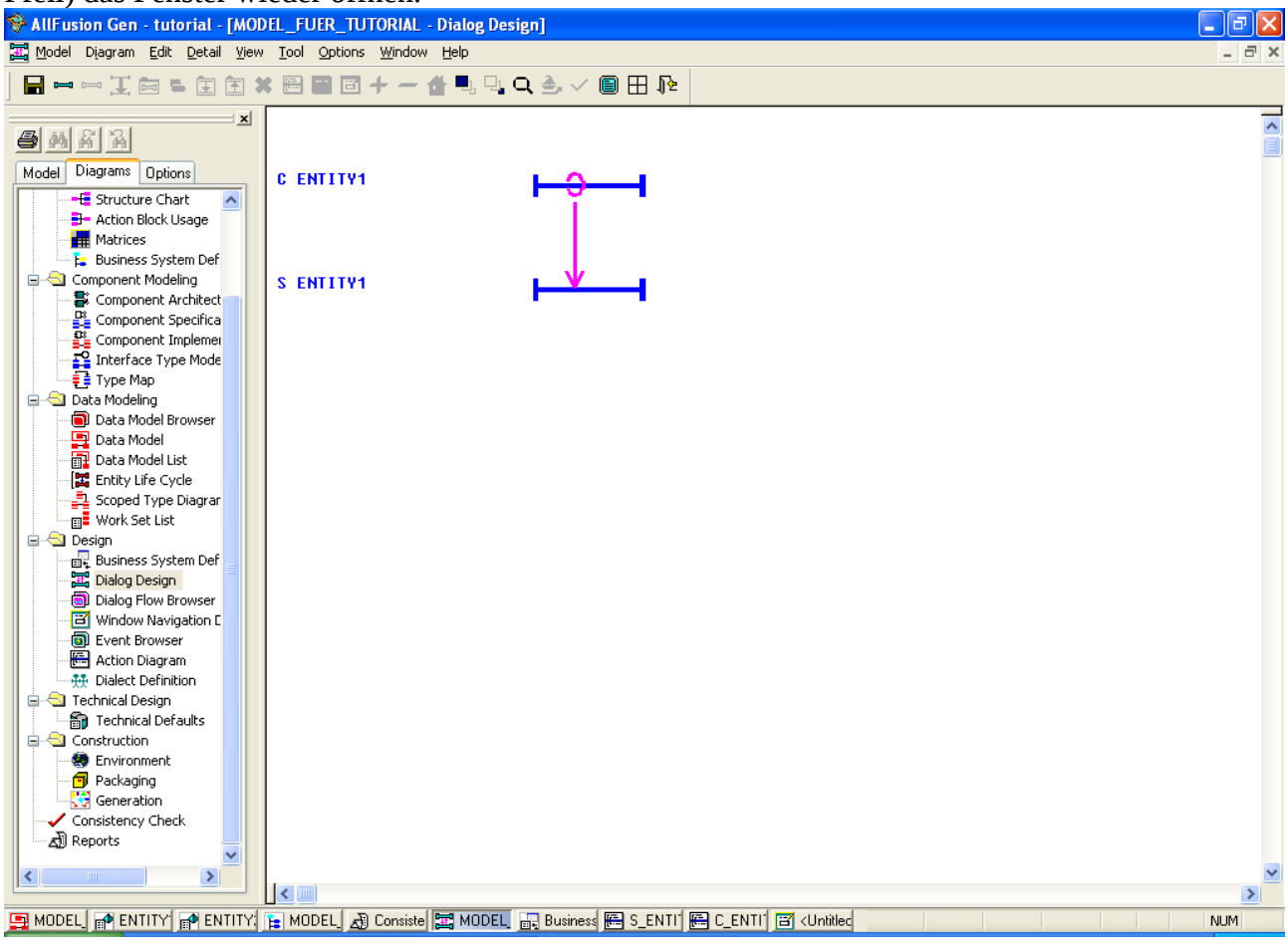


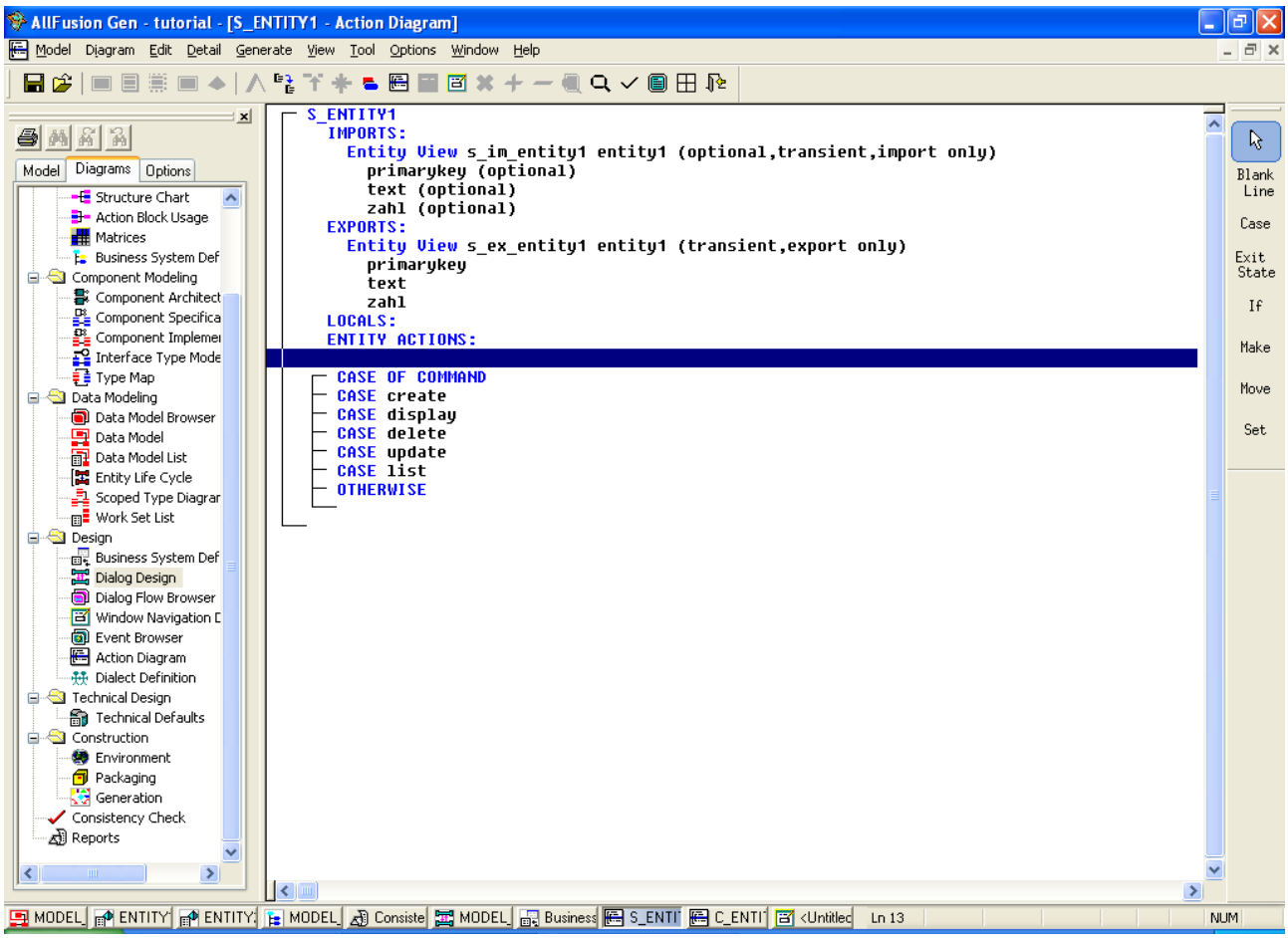
Als nächstes müssen noch die Import- und Exportviews der beiden Dialoge verknüpft werden. Wählen Sie zunächst die Option „Data Sent“. Klicken Sie auf die Importview vom Serverdialog und anschließend auf die passende Exportview vom Clientdialog. Dann drücken Sie auf „Match“.

Das selbe tun Sie dann bei „Data Returned“.

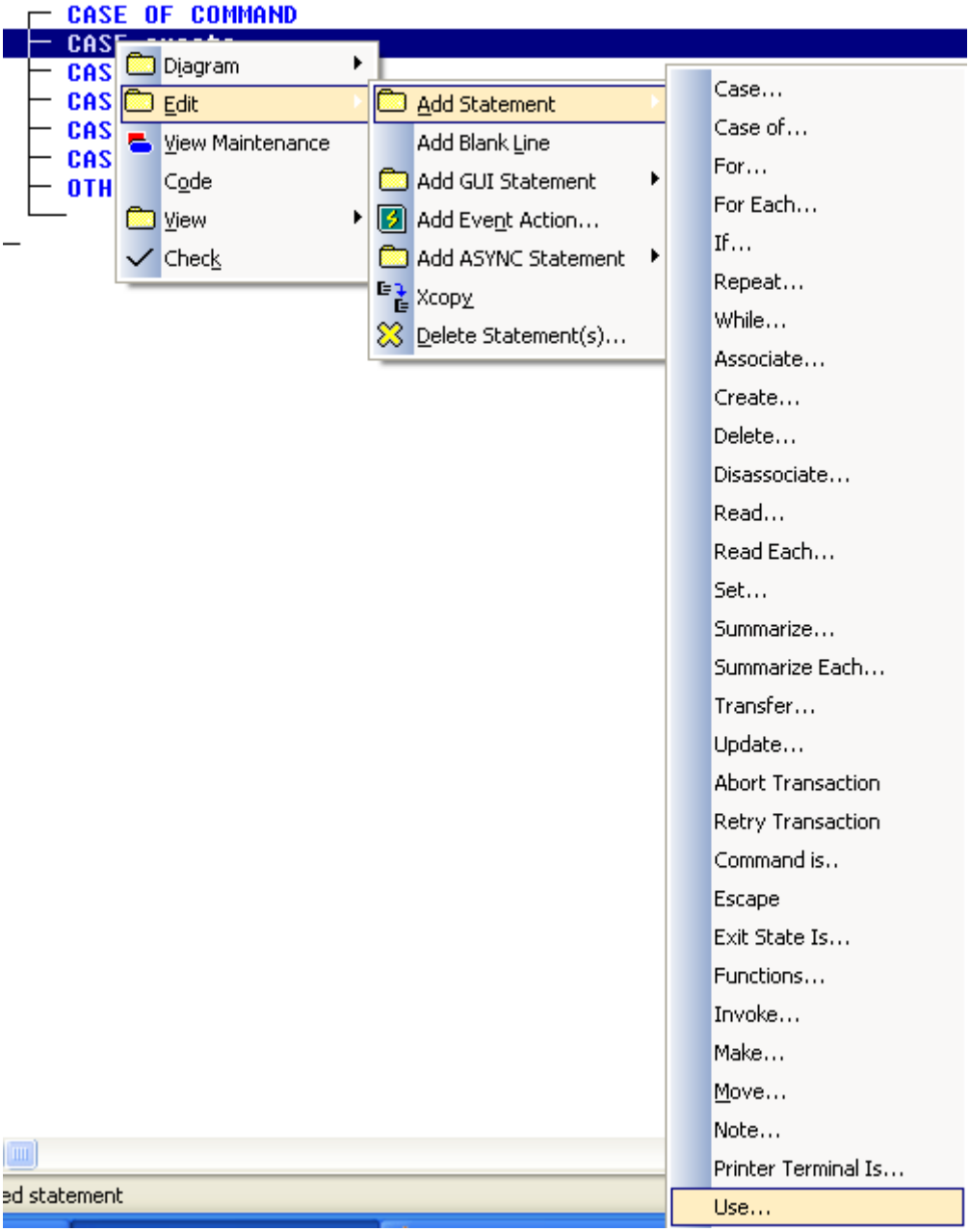


Unter „Sent/Returned“ können Sie sich die Übersicht über alle existierenden Views und die Verknüpfungen ansehen. Wenn alle Einstellungen gemacht wurden, drücken Sie auf OK.
Sie können den bestehenden Join auch noch ändern, indem Sie über Doppelklick auf den Join (rosa Pfeil) das Fenster wieder öffnen.

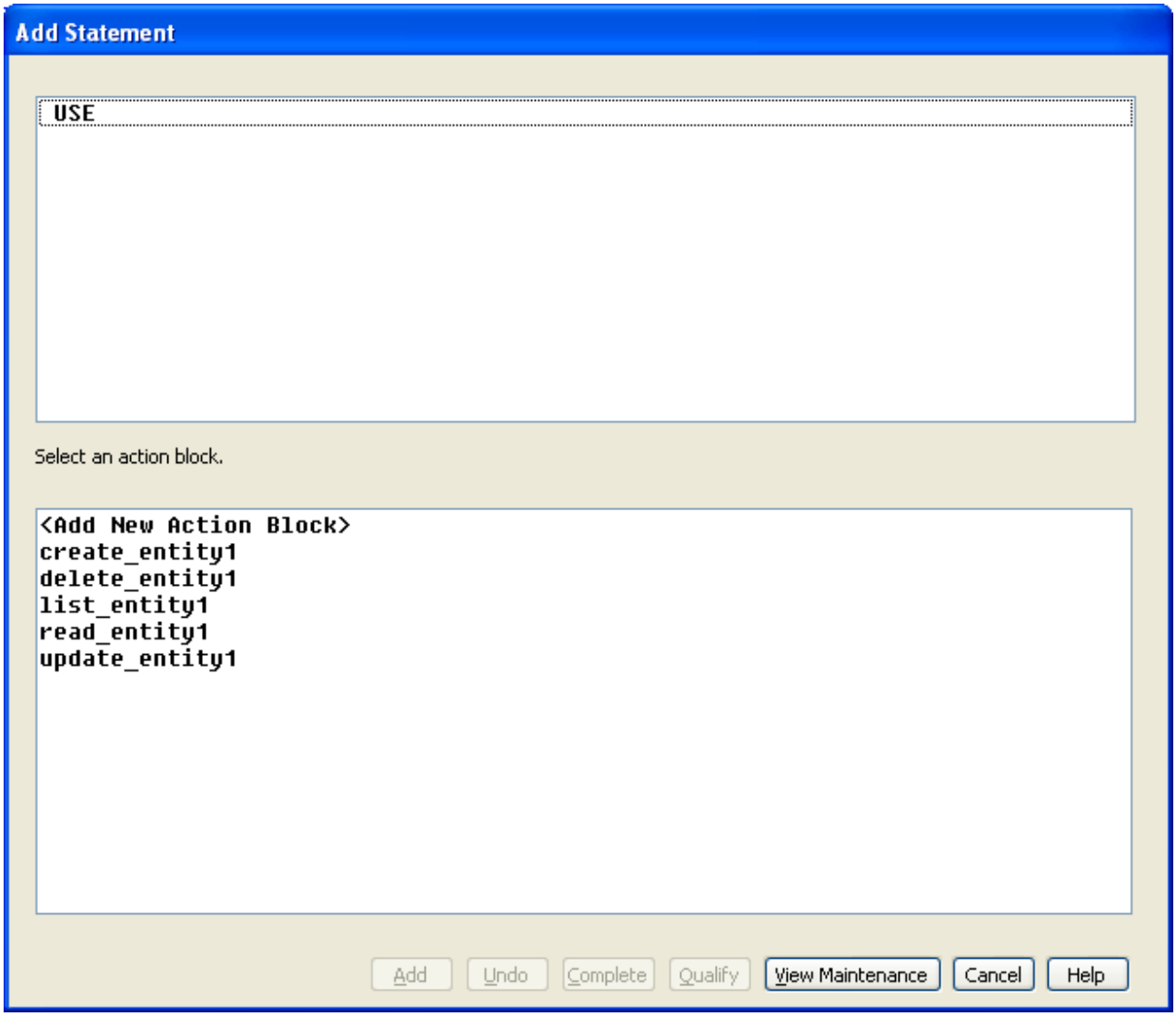




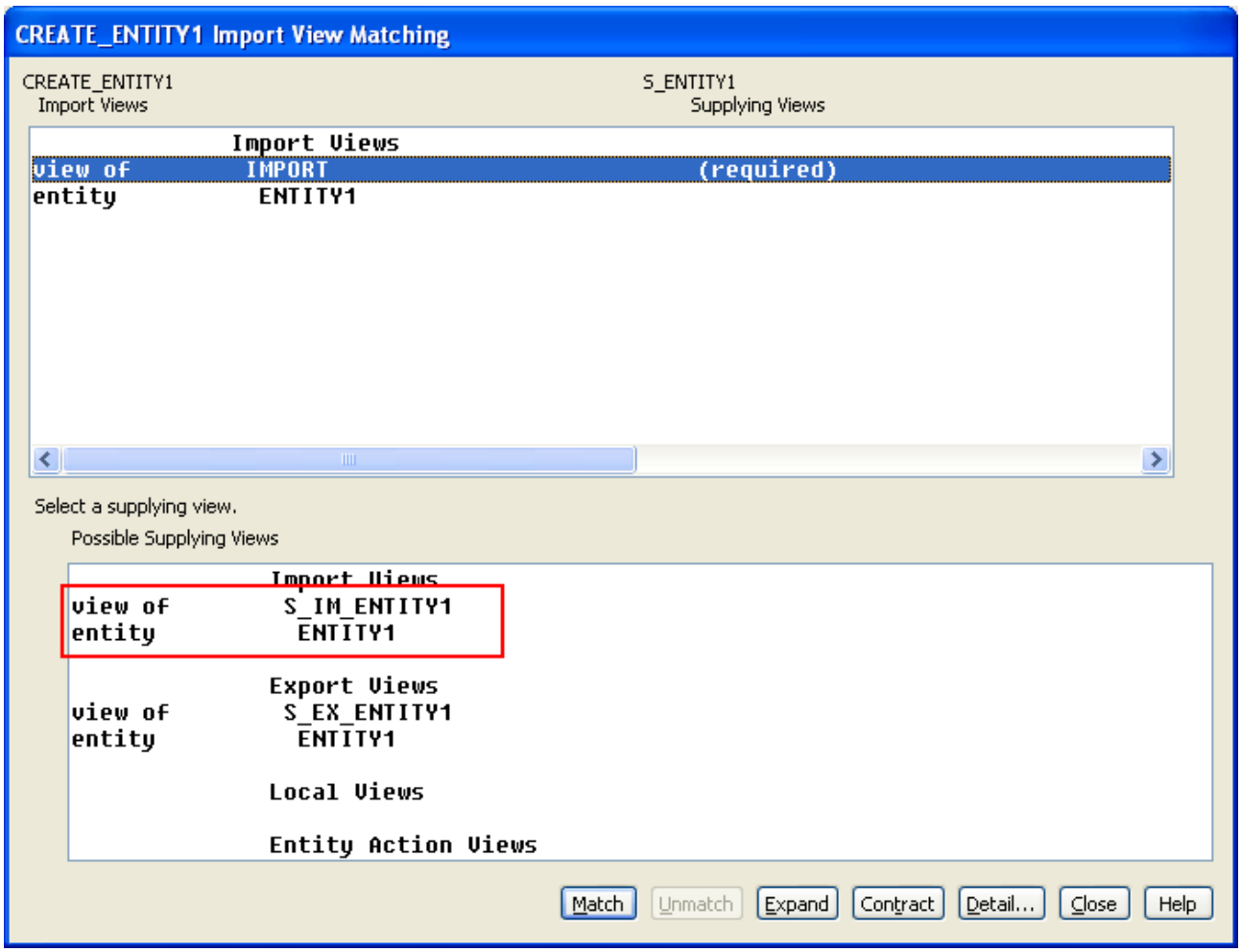
Nun geht’s in den Action Block des Servers. Legen Sie auch hier einen „Case of command“ an wie im Client. Zu jedem „Case“ fügen Sie nun ein „Use“ Statement hinzu.



„Use“ ist quasi ein Funktionsaufruf. Mit diesem Statement können die Elementarprozesse aus der Activity Hierachy aufgerufen werden.

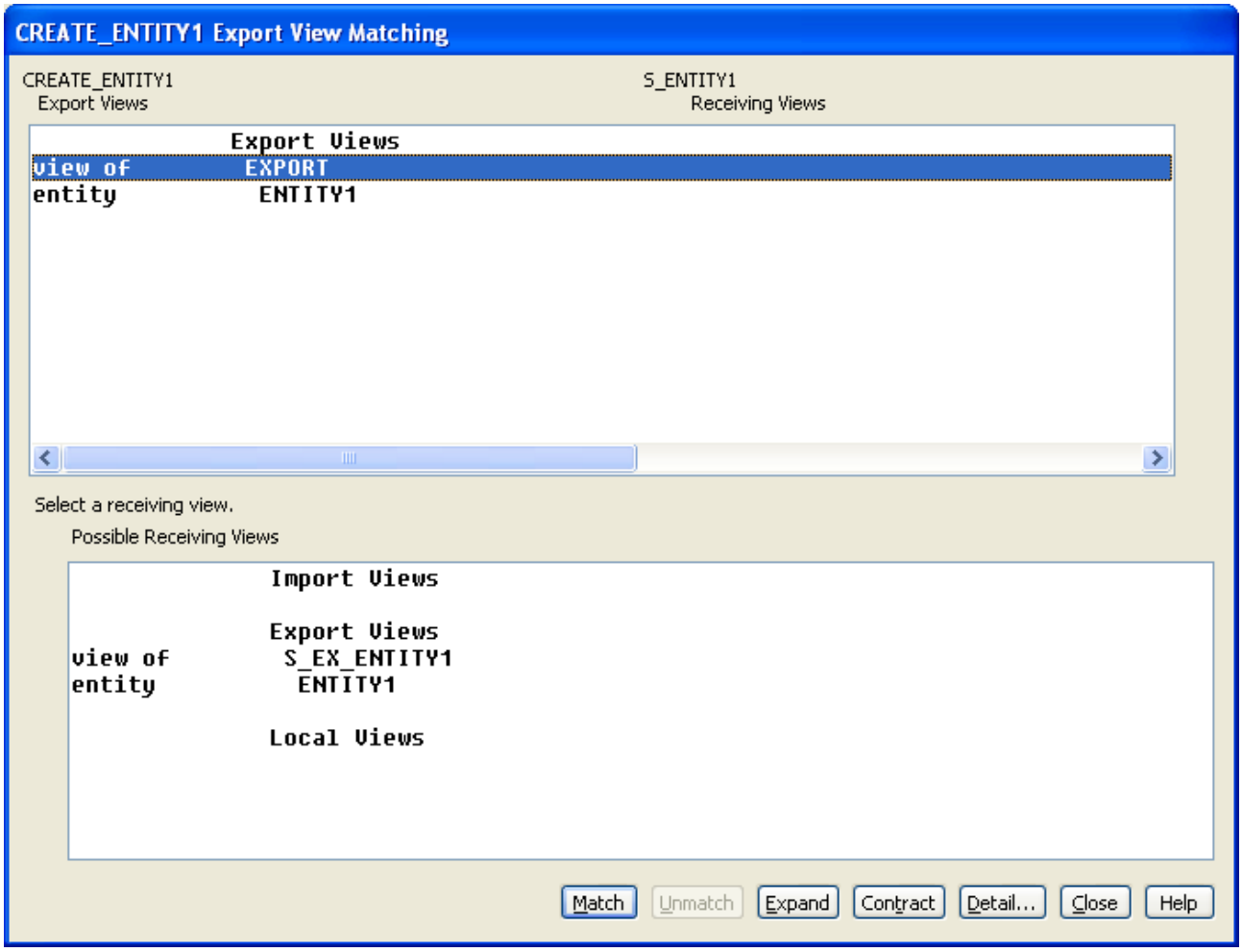


Wählen Sie für jeden Befehl den entsprechenden Elementarprozess aus und drücken sie auf „Add“.

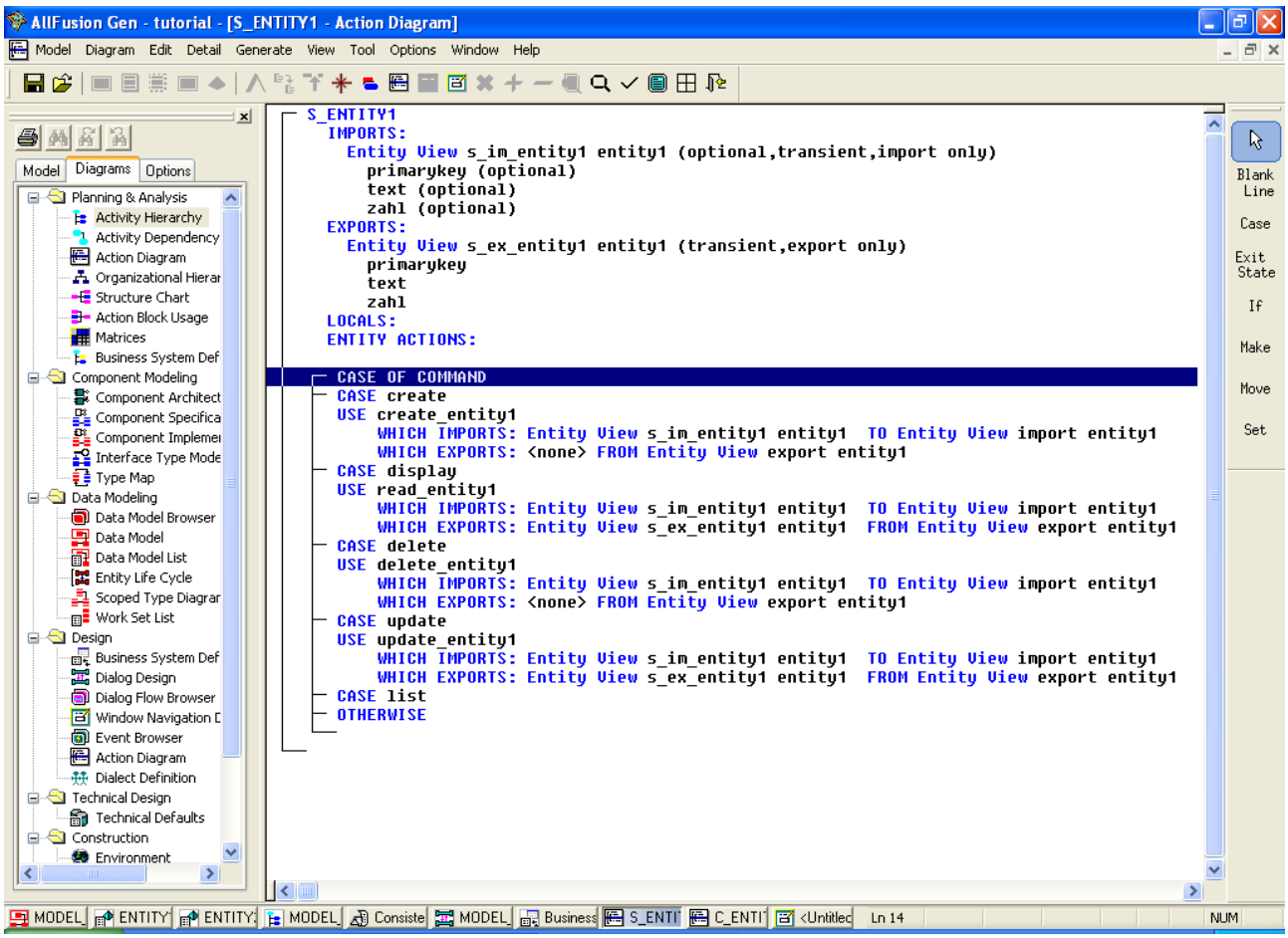


Nun wird sich ein Fenster öffnen, in welchem Sie zunächst angeben aus welcher View die Eingabedaten für den Elementarprozess stammen sollen (hier bei „Create“). Die Anmerkung „required“ bedeutet, dass hier eine Datenzufuhr zwingend notwendig ist.

Wählen Sie zuerst die Zeile mit „view of“ aus, und anschließend eine passende Zeile aus der Auswahl unten. Drücken Sie anschließend auf „Match“. Danach können Sie auf „Close“ drücken.



Wenn Sie das Fenster für den Import geschlossen haben, wird sich das Fenster für den Export öffnen. Sie können hier genauso verfahren wie beim Import, jedoch ist es nicht zwingend notwendig hier eine View anzugeben. Wenn einfach auf Close drücken, werden sich die Eingabefelder wieder leeren wenn Sie auf „Create“ drücken, da die Daten von einer leeren View überschrieben werden.



In diesem Bild wurden alle Elementarprozesse bis auf List eingefügt. List benötigt noch ein paar weitere Views, und wird erst im nächsten Kapitel behandelt.

Hinweis: Wann man den Export eines „Use“ Befehls weglassen kann, hängt vom Elementarprozess ab. Darum hier eine kurze Übersicht über alle Elementarprozesse und ihre eigentliche Funktionsweise:

Create – Export kann ignoriert werden! Aus der Importview wird eine neue Zeile in der entsprechenden Datenbanktabelle erstellt. Nachdem die Datenbank gefüllt wurde, wird der Inhalt in den Export kopiert und wieder zurückgegeben.

Read – Export kann **nicht** ignoriert werden! Read benötigt keine vollständig befüllte View. Alle Attribute bis auf den Primärschlüssel werden ignoriert. Read sucht in der Datenbanktabelle den Eintrag mit dem angegebenen Primärschlüssel und erstellt dann als Export einen View, der alle Daten dieser Zeile beinhaltet.

Update – Export kann ignoriert werden! Funktioniert ähnlich wie Create. Es wird der Eintrag gesucht mit dem angegebenen Primärschlüssel und anschließend werden alle Attribute mit den übergebenen überschrieben. Man kann den Primärschlüssel nicht mit Update ändern.

Delete – Export kann ignoriert werden! Ähnlich wie bei Read wird nur der Primärschlüssel benötigt. Der entsprechende Eintrag in der Datenbanktabelle wird dann gelöscht, aber vorher noch

List – Export kann **nicht** ignoriert werden! Gibt die gesamte Datenbanktabelle zurück. Hat keine Eingabe.