

# Event Code

The event editor is very flexible and may initially look daunting. Events are put together with 'code'. However, the structure of the code is relatively simple and the code is written with buttons so you do not get syntax errors. The manual describes that different checks and events. The following shows how these events and check are used in combination.

## Notifying a player of an event:

Prep: You have to know the number of the regimes you want to send the message to. This is usually straight forward. However, some scenarios have many regimes so you may have to go to the regime tab to get the exact number (e.g., Poland is regime 9 in the WWII scenario). Also, if you want to include variable names in the events, you need to know the number of the variable. This is in the setting tab - game variable tab. If you want a picture to go with the event, you must have the picture attached to the scenario. To do this, go to the setting tab - event picture tab. Remember the number of the picture

Code:

```
exec message 2 (0,1,-1,-1) (Hi there)
```

or

```
loop tempvar 0 to 9  
exec message (tempvar 0 , -1, tempvar 0, -1) (Hi there regime [tempvar]0[/tempvar])  
end loop
```

or

```
setvar victorypoint (2) = checkvp(0)  
exec message (0,-1,-1,-1) (you have [gamevar]2[/gamevar] victory points)
```

## Comments

The first event will send a message to regime 0 and regime 1. There will be no picture and the scenario description will not change. The message will say 'Hi there'.

The second event uses a loop. This is useful if you want to send a message to more than 2 regimes. There is also an embedded variable (tempvar). Regime 2 would see 'Hi there regime 2'. Each regime would also get a different picture for the event (e.g., regime 3 gets event picture number 3).

The third event sends a game variable to player 0. In this case, the number of victory points. You can expose variables to all players in the game var panel (under game settings). However, they would be visible to all players. This event allows you to selectively expose a variable to a player (e.g., in the WWII scenario, the Western Allied player sees how many u-boats the German player will deploy next turn if Ultra is cracked during the previous turn).

If you want to add a picture to any of the above, you have to set up the link to the picture. In the editor, go to 'Settings' and then select 'Event Pictures'.

If you ever have questions about the arguments in the check or the event, do a mouse over or check the manual.

In general, the message exec is used the most often. Messages get logged in the reports tab. This is helpful for players to keep track of what is going on. When writing events, it is often useful to add execs to see how the flow of your events is implemented.

## How events are called during the game

An event is called or checked one of three ways: No Check, Turn Check, and Round Check. The default for a new event is No Check. You can change how the event is called by selecting a line in the event and then toggling a button outside the event box.

No Check: This event is normally skipped. The event can only be called by an action card or another event. This is how action cards are normally implemented. *If your event is not working, the first thing to check is if it is properly called (e.g., most likely a 'No Check')*

Turn Check: In AT, each player gets a turn every round. Turn checks allow you to immediately do something without letting the other players respond (e.g., country surrenders immediately if it's capital is occupied. Turn checks execute at the beginning of each turn (so a turn check on turn 0 is equivalent to a round check).

Round Checks: These occur at the beginning of a round and are the most common type of event. A round is the collection of everyone's turn.

## Conditional statements

Logical statements can occur in any type of event (e.g., No Check, Turn Check, Round Check). Here is how you could structure of different types of conditionals.

**If - then:** This is the basic conditional:

```
Check (something) =>< (something)
Any combination of loops, checks, setvars, and execs
End.Ch
```

**If Not - then:** AT does equals, less than, greater than, but does have 'not equal to'. You can implement like this:

```
setvar tempvar0=0
Check ()=()
setvar tempvar0=1
End.Ch
Check tempvar0=0
Execute not statements
End.Ch
```

**While Do:** A bit trickier. I used this to randomly assign territories during the 'Pickem Scenario'

```
loop tempvar0=0 to 2
setvar tempvar0=0
Check () = ()
setvar tempvar0=2
execute statements
End.Ch
execute other statements
End.Loop
```

**Check both:**

```
Check condition a
Check condition b
...
End.ch
End.ch
```

**Check either:**

```
setvar tempvar0=0

Check condition a
setvar tempvar0=1
End.ch

Check condition b
setvar tempvar0=1
End.ch

Check tempvar0==1
...
End.ch
```

**Check neither:**

```
setvar tempvar0=0

Check condition a
setvar tempvar0=1
End.ch

Check condition b
setvar tempvar0=1
End.ch

Check tempvar0==0
...
End.ch
```

*Note that tempvar is a throw away variable that you can use within an event. Gamevars are stored by the game and work across events.*

## Common Loops

Loops are nice ways to write events with less lines. Here are some that are used in several scenarios:

Deal action cards to every player. This is a good way to initially deal lots of cards.

```
Looper tempvar0= 0 to # of players
Looper tempvar1= 0 to # of action cards
ExecGiveActionCard (tempvar1, tempvar0)
End Loop
End Loop
```

Check the map. This will put a partisan on every hex that has been define by slot (1,1). In ACW, I defined confederate rail hexes this way. If they were occupied by the Union, their was a chance that a Confederate raider would arrive and cut the rail line.

```
Looper x=0 to checkmapwidth
Looper y=0 to checkmapheight
Checkslot (1, x, y) = 1 - this slot (group of hexes) could represent pre-war Russia
Checkowner (x,y) = 0 - checking to see if pre-war Russia hex now occupied by Germans
Checkrandom < Partisan_level - may tailor % chance of partisans based on game activity code to add units
End Check
End Check
End Check
End Loop
End Loop
```

## WWII Scenario Event Walk Though.

The next couple of sections go through events in the War in Europe scenario. This was written about 6 months ago and the scenario has been revised since then, so the actual events may differ. However, the purpose of this is to give a feel as to how checks and events can be put together.

The first 12 events execute 'set unit arrival' action card. Note that all action cards must link to an event (usually a 'no check type) to do anything meaningful. Each of the 3 major powers (German, West, and Soviet Union) have action cards to set the arrival of units that are purchased with other action cards. This gives players more flexibility. Also, separating the unit from the arrival is much easier for the scenario developer. 3 players have 4 potential locations where they can purchase about 12 different units. Separating the arrival from the unit purchase means you only have to implement 12 + 12 cards vs 12 x 12 cards. *Since the time of the writing, I took out the German arrival card in Paris and Kiev*

Here is the code of the first event. The next 11 are basically the same with the message, variables, and card given varying. Note also that the event ends with the action card being given back to the player. This allows the player to play the action card multiple times. Also, the message is nice because it allows the player to keep track of changes that are made

```
0) EXECUTE: ExecMessage2(0, -1, -1, -1)
1) SETVAR: Gameslot_German X(#0) = 60 I use a gamevar here so the game remembers where this was set
2) SETVAR: Gameslot_German Y(#1) = 21
3) EXECUTE: ExecGiveActionCard(1, 0)
```

The next event deals out the cards. Note that I start out with a looper since the cards run in sequence (player 0 gets cards 0-3, 1 gets 4-7, 3 gets 8-11). I use 3 different variables so I only have to run the looper once. The first step is to remove the cards. I do this since I do not know if the player has lost the arrival hex (e.g., did the West lose Paris?) the previous turn. Once the players hand is clear, I deal the cards if the player controls the hex (e.g., if Germany controls Rome, they can set the arrival there). I repeat this check 12 times (not shown but similar to lines 9-11)

```
0) LOOPER: TempVar0 FROM 0 TO 3
1) SETVAR: TempVar1 = TempVar0
2) SETVAR: TempVar1 + 4
3) SETVAR: TempVar2 = TempVar0
4) SETVAR: TempVar2 + 8
5) EXECUTE: ExecRemoveActionCard(TempVar0, 0)
6) EXECUTE: ExecRemoveActionCard(TempVar1, 1)
7) EXECUTE: ExecRemoveActionCard(TempVar2, 2)
8) END LOOPER
9) CHECK: CheckHexOwner(39, 17) == 0
10) EXECUTE: ExecGiveActionCard(0, 0)
11) END CHECK
12) CHECK: CheckHexOwner(60, 21) == 0
13) EXECUTE: ExecGiveActionCard(1, 0)
14) END CHECK
15) ...
```

The next 16 events execute the purchase unit action cards. I need 1 event per unit card in the deck. I start with a check turn so I know which player played the card. This then allows me to set the correct unit nationality and set the arrival to the correct hex. After the unit arrives, the card is given back to the player. The player can then play the card as many times as desired, until he runs out of pp to spend. Note that the unit number (first argument, must be pre-defined in the units tab of the editor).

```
0) CHECK: CheckTurn == 0
1) EXECUTE: ExecAddUnit(0, Gameslot_German X(#0), Gameslot_German Y(#1), 0)
2) END CHECK
3) CHECK: CheckTurn == 1
4) EXECUTE: ExecAddUnit(0, Gameslot_Western X(#2), Gameslot_Western Y(#3), 1)
5) END CHECK
6) CHECK: CheckTurn == 2
7) EXECUTE: ExecAddUnit(0, Gameslot_SU X(#4), Gameslot_SU Y(#5), 2)
8) END CHECK
9) SETVAR: TempVar0 = CheckTurn
10) EXECUTE: ExecGiveActionCard(12, TempVar0)
11) EXECUTE: ExecMessage2(TempVar0, -1, -1, -1)
```

Here is the initial deal of unit action cards. I use loopers to minimize the lines of code I need. I loop through the action cards and then the players (German is 0, West is 1, Soviets are 2)

```
0) CHECK: CheckRound == 1
1) LOOPER: TempVar0 FROM 12 TO 22
2) LOOPER: TempVar1 FROM 0 TO 2
3) EXECUTE: ExecGiveActionCard(TempVar0, TempVar1)
4) END LOOPER
5) END LOOPER
6) END CHECK
```

I deal special unit cards later in the war. I defined some special units (SS/ shock troops and elite airbourne). This event deals them out at the appropriate time and to the appropriate player.

```
0) CHECK: CheckYear == 1942
1) CHECK: CheckMonth == 1
2) EXECUTE: ExecGiveActionCard(23, 0)
3) EXECUTE: ExecGiveActionCard(24, 0)
4) EXECUTE: ExecGiveActionCard(25, 0)
5) EXECUTE: ExecGiveActionCard(26, 0)
6) EXECUTE: ExecGiveActionCard(26, 2)
7) EXECUTE: ExecGiveActionCard(25, 2)
8) EXECUTE: ExecGiveActionCard(24, 2)
9) EXECUTE: ExecGiveActionCard(23, 2)
10) EXECUTE: ExecMessage2(0, -1, -1, -1)
11) EXECUTE: ExecMessage2(2, -1, -1, -1)
12) END CHECK
13) END CHECK
14) CHECK: CheckYear == 1944
15) CHECK: CheckMonth == 1
16) EXECUTE: ExecGiveActionCard(27, 1)
17) EXECUTE: ExecMessage2(1, -1, -1, -1)
18) END CHECK
19) END CHECK
```

Make ships go faster. I thought the base ship speed was too low given the time and scale of the scenario. The ships are movement type 8 and they move at quadruple speed (or 25% of the cost). Note that I could have also changed the speed of the units in the masterfile. In general, if you want to make minor tweaks to the masterfile, it is probably best to do that with events. If you want to make major tweaks, you should think about a new masterfile. However once you use a new masterfile, you have to support it yourself.

```
0) EXECUTE: ExecMveTypeModifier(8, 25)
```

This next set of events sets political point production above what is available on the map. I use this to simulate US entry, Soviet off map production. Also, this production can be 'attacked' through 'strategic' warfare. The next few events set variables that will be used later in calculating the total pp each major power gets.

If you go to the settings tab, then game variables tab, you can see all the variables that are defined as well as there initial values. You must pre-define a variable before you can use it. Pre-defining variables is done in the editor - setting - game variables tab.

This sets the bonuses available to the West and the Soviet Union later in the war:

```
0) CHECK: CheckYear > 1942
1) SETVAR: Gameslot_US post 1943(#16) = 50
2) EXECUTE: ExecMessage2(1, -1, -1, -1)
3) SETVAR: Gameslot_SU post 1943(#20) = 50
```

- 4) EXECUTE: ExecMessage2(2, -1, -1, -1)
- 5) EXECUTE: BlockEvent
- 6) END CHECK

This does the same for Germany ("Speer's reforms")

- 0) CHECK: CheckYear > 1943
- 1) SETVAR: Gameslot\_Germany Post 1944(#11) = 50
- 2) EXECUTE: ExecMessage2(0, -1, -1, -1)
- 3) END CHECK
- 4) EXECUTE: BlockEvent

This gives Germany 10 pp if they control Oslo ("Swedish Iron Ore flows")

- 0) CHECK: CheckHexOwner(37, 3) == 0
- 1) SETVAR: Gameslot\_Swedish Iron Ore(#6) = 10
- 2) END CHECK
- 3) CHECK: CheckHexOwner(37, 3) > 0
- 4) SETVAR: Gameslot\_Swedish Iron Ore(#6) = 0
- 5) END CHECK

The 'US' gives the West lend lease aid after the fall of Paris.

- 0) CHECK: Gameslot\_Fall of France(#9) == 1
- 1) CHECK: Gameslot\_US Lend Lease(#14) == 0
- 2) SETVAR: Gameslot\_US Lend Lease(#14) = 25
- 3) EXECUTE: ExecMessage2(1, 0, -1, -1)
- 4) END CHECK
- 5) END CHECK

This is the 'US' basic contribution to the war effort in Europe. Starts in Jan of '42

- 0) CHECK: CheckYear > 1941
- 1) CHECK: Gameslot\_US Entry(#15) == 0
- 2) SETVAR: Gameslot\_US Entry(#15) = 75
- 3) EXECUTE: ExecMessage2(0, 1, -1, -1)
- 4) END CHECK
- 5) END CHECK

The Soviet PP production jumps from 10 (preset) to 50 (or +40) when Germany invades

- 0) CHECK: CheckWar(0, 2) == 1
- 1) CHECK: Gameslot\_SU war with Germany(#19) == 0
- 2) SETVAR: Gameslot\_SU war with Germany(#19) = 40
- 3) EXECUTE: ExecMessage2(0, 2, -1, -1)
- 4) END CHECK
- 5) END CHECK

Here, I define Winter as December through January. I do this now because I will not allow Soviet lend lease aid during the Winter as it arrives in the Artic ports.

- 0) SETVAR: Gameslot\_Is Winter(#10) = 0
- 1) CHECK: CheckMonth == 12
- 2) SETVAR: Gameslot\_Is Winter(#10) = 1
- 3) END CHECK
- 4) CHECK: CheckMonth == 1
- 5) SETVAR: Gameslot\_Is Winter(#10) = 1
- 6) END CHECK
- 7) CHECK: CheckMonth == 2
- 8) SETVAR: Gameslot\_Is Winter(#10) = 1
- 9) END CHECK

Here I give the West the ability to 'lend' 20 pp to the Soviet Union (note that the card costs 25 pp so there is some wastage). I take away the card first and then give it to the West only if it is not Winter and the SU and Germany are at war. Sequencing of events is important. Notice that I determine whether the turn is Winter before I have the events that are impacted by winter.

- 0) EXECUTE: ExecRemoveActionCard(28, 1)
- 1) CHECK: CheckWar(0, 2) == 1
- 2) CHECK: Gameslot\_Is Winter(#10) == 0
- 3) EXECUTE: ExecGiveActionCard(28, 1)
- 4) END CHECK
- 5) END CHECK

This executes the SU lend lease action card. Note that the card is given back after played so the West can give as much lend lease as it can afford.

- 0) EXECUTE: ExecGiveRegimePP(2, 20)
- 1) EXECUTE: ExecMessage2(1, 2, -1, -1)
- 2) EXECUTE: ExecGiveActionCard(28, 1)

This calculates all the pp points that are granted at the start of the round. Each country gets the base plus all modifiers that are set in the events above. Notice how the operators work. = means equals, + means add to, - means subtract from, \* means multiply by, and / means divide by.

- 0) SETVAR: Gameslot\_German PP(#13) = Gameslot\_German Base Production(#12)
- 1) SETVAR: Gameslot\_German PP(#13) + Gameslot\_Germany Post 1944(#11)
- 2) SETVAR: Gameslot\_German PP(#13) + Gameslot\_Axis Minors in War(#8)
- 3) SETVAR: Gameslot\_German PP(#13) + Gameslot\_Italy in War(#7)
- 4) SETVAR: Gameslot\_German PP(#13) + Gameslot\_Swedish Iron Ore(#6)
- 5) EXECUTE: ExecGiveRegimePP(0, Gameslot\_German PP(#13))
- 6) SETVAR: Gameslot\_Western PP(#18) = Gameslot\_Western Base Production(#17)
- 7) SETVAR: Gameslot\_Western PP(#18) + Gameslot\_US post 1943(#16)
- 8) SETVAR: Gameslot\_Western PP(#18) + Gameslot\_US Entry(#15)
- 9) SETVAR: Gameslot\_Western PP(#18) + Gameslot\_US Lend Lease(#14)
- 10) EXECUTE: ExecGiveRegimePP(1, Gameslot\_Western PP(#18))
- 11) SETVAR: Gameslot\_Soviet PP(#22) = Gameslot\_SU Base Production(#21)
- 12) SETVAR: Gameslot\_Soviet PP(#22) + Gameslot\_SU post 1943(#20)
- 13) SETVAR: Gameslot\_Soviet PP(#22) + Gameslot\_SU war with Germany(#19)
- 14) EXECUTE: ExecGiveRegimePP(2, Gameslot\_Soviet PP(#22))

The next set of events are used to simulate strategic warfare. I decided to abstractly represent the u-boat and bomber campaigns. I set up 4 'strategic' units – Uboats and Allied bombers are offensive, DD escorts and German interceptors are defensive. The units are really just game variables that the player can purchase with action cards.

This first event calculates UBoat damage to PP. The Germans do 0-2 points of damage per UBoat each turn. They do an extra point during the first 6 months of 1942 (due to US dis-organization). They also get an extra point if they control 1 of the French ports on the Atlantic and an extra point if they control the Norwegian North Sea port (both due to having better bases). They end of the turn makes sure that the UBoat damage does not exceed the number of Western PP earned during the turn. I also make the Uboat damage variable negative and then take the PPs away from the west.

- 0) SETVAR: Gameslot\_Uboat pp damage(#31) = 0 *I reset the damage counter*
- 1) CHECK: Gameslot\_U-Boats(#27) > 0 *If there is more than 1 uboat, I now calculate the damage*
- 2) SETVAR: TempVar0 = 0
- 3) CHECK: CheckHexOwner(17, 31) == 0 *I check to see if Germany owns at least 1 French Atlantic port*
- 4) SETVAR: TempVar0 = 1
- 5) END CHECK
- 6) CHECK: CheckHexOwner(18, 29) == 0 *I check to see if Germany owns at least 1 French Atlantic port*
- 7) SETVAR: TempVar0 = 1
- 8) END CHECK
- 9) CHECK: CheckHexOwner(18, 24) == 0 *I check to see if Germany owns at least 1 French Atlantic port*
- 10) SETVAR: TempVar0 = 1
- 11) END CHECK
- 12) CHECK: CheckHexOwner(16, 21) == 0 *I check to see if Germany owns at least 1 French Atlantic port*
- 13) SETVAR: TempVar0 = 1
- 14) END CHECK
- 15) CHECK: CheckHexOwner(33, 1) == 0
- 16) SETVAR: TempVar0 + 1
- 17) END CHECK
- 18) CHECK: CheckYear == 1942
- 19) CHECK: CheckMonth < 7
- 20) SETVAR: TempVar0 + 1
- 21) END CHECK
- 22) END CHECK
- 23) LOOPER: TempVar1 FROM 1 TO Gameslot\_U-Boats(#27) *This is where each uboat is firing*
- 24) SETVAR: TempVar2 = CheckRandomPercent
- 25) SETVAR: TempVar2 / 33
- 26) SETVAR: TempVar2 + TempVar0
- 27) SETVAR: Gameslot\_Uboat pp damage(#31) + TempVar2
- 28) END LOOPER
- 29) CHECK: Gameslot\_Uboat pp damage(#31) > Gameslot\_Western PP(#18) *Damage cannot exceed total grant*
- 30) SETVAR: Gameslot\_Uboat pp damage(#31) = Gameslot\_Western PP(#18)
- 31) END CHECK
- 32) SETVAR: TempVar3 = Gameslot\_Uboat pp damage(#31)
- 33) SETVAR: TempVar3 \* -1
- 34) EXECUTE: ExecGiveRegimePP(1, TempVar3)
- 35) END CHECK

This event calculates Uboat damage to DD escorts. Uboats are relatively ineffective here doing 0-1 points of damage per boat. At the end, I make sure that the damage to DD escorts does not exceed the number of DD escorts (e.g., the West cannot have a negative number of escorts)

```
0) SETVAR: Gameslot_Uboat DD damage(#32) = 0
1) CHECK: Gameslot_U-Boats(#27) > 0
2) LOOPER: TempVar1 FROM 1 TO Gameslot_U-Boats(#27)
3) SETVAR: TempVar2 = CheckRandomPercent
4) SETVAR: TempVar2 / 50
5) SETVAR: Gameslot_Uboat DD damage(#32) + TempVar2
6) END LOOPER
7) CHECK: Gameslot_Uboat DD damage(#32) > Gameslot_DD Escorts(#28) Can't sink more dd's than exist
8) SETVAR: Gameslot_Uboat DD damage(#32) = Gameslot_DD Escorts(#28)
9) END CHECK
10) END CHECK
```

This is the damage that Western DD's do to Uboats. Base amount is 1-3, +1 if ultra is broken the previous turn (this is accomplished based on the order of events. Ultra is reset towards the end of the event list. The West also gets an extra point in 1943 due to the use of light aircraft carriers in the escort role.

```
0) SETVAR: Gameslot_DD damage(#33) = 0
1) CHECK: Gameslot_DD Escorts(#28) > 0
2) SETVAR: TempVar0 = 0
3) CHECK: CheckYear > 1942
4) SETVAR: TempVar0 = 1
5) END CHECK
6) LOOPER: TempVar1 FROM 1 TO Gameslot_DD Escorts(#28)
7) SETVAR: TempVar2 = CheckRandomPercent
8) SETVAR: TempVar2 / 33
9) SETVAR: TempVar2 + 1
10) SETVAR: TempVar2 + TempVar0
11) SETVAR: TempVar2 + Gameslot_Ultra(#38)
12) SETVAR: Gameslot_DD damage(#33) + TempVar2
13) END LOOPER
14) CHECK: Gameslot_DD damage(#33) > Gameslot_U-Boats(#27)
15) SETVAR: Gameslot_DD damage(#33) = Gameslot_U-Boats(#27)
16) END CHECK
17) END CHECK
```

This event debits both the Uboat and DD escort variables based on the damage from the previous events. I also present the 'Uboat Report' to both Germany and the West. Each side now knows what the other did the previous turn. If the West breaks Ultra, they will know what the Germans will deploy in the next turn so they can react.

```
0) CHECK: Gameslot_U-Boats(#27) > 0
1) EXECUTE: ExecMessage2(0, 1, -1, -1)
2) END CHECK
3) SETVAR: Gameslot_U-Boats(#27) - Gameslot_DD damage(#33)
4) SETVAR: Gameslot_DD Escorts(#28) - Gameslot_Uboat DD damage(#32)
```

The next 2 events execute the action cards that allow Germany and the West to buy UBoats and DD escorts. Here is the Uboat event, the DD event is similar

```
0) SETVAR: Gameslot_U-Boats(#27) + 5
1) EXECUTE: ExecGiveActionCard(29, 0)
2) EXECUTE: ExecMessage2(0, -1, -1, -1)
```

Here I deal out the initial strategic action cards

```
0) CHECK: CheckRound == 1
1) EXECUTE: ExecGiveActionCard(30, 1)
2) EXECUTE: ExecGiveActionCard(29, 0)
3) EXECUTE: ExecGiveActionCard(31, 1)
4) EXECUTE: ExecGiveActionCard(32, 0)
5) END CHECK
```

The next few event calculate and then apply bomber pp and interceptor damage and interceptor damage to bombers. They are basically the same as the U-Boat / DD events with different modifiers. I show the Bomber PP damage event below. Bombers do 0-2 PP of damage which increases by 1 once the US enters the war due to daylight bombing.

```
0) SETVAR: Gameslot_Bomber pp damage(#34) = 0
1) CHECK: Gameslot_Bombers(#29) > 0
2) SETVAR: TempVar0 = 0
3) CHECK: CheckYear > 1941
4) SETVAR: TempVar0 = 1
5) END CHECK
6) LOOPER: TempVar1 FROM 1 TO Gameslot_Bombers(#29)
```

```

7) SETVAR: TempVar2 = CheckRandomPercent
8) SETVAR: TempVar2 / 33
9) SETVAR: TempVar2 + TempVar0
10) SETVAR: Gameslot_Bomber pp damage(#34) + TempVar2
11) END LOOPER
12) CHECK: Gameslot_Bomber pp damage(#34) > Gameslot_German PP(#13)
13) SETVAR: Gameslot_Bomber pp damage(#34) = Gameslot_German PP(#13)
14) END CHECK
15) SETVAR: TempVar3 = Gameslot_Bomber pp damage(#34)
16) SETVAR: TempVar3 * -1
17) EXECUTE: ExecGiveRegimePP(0, TempVar3)
18) END CHECK

```

The next events deal with the fall of France. The variable here will be used to set other down stream events in motion. As you can see from earlier events, the 'lend lease' pp bonus is triggered. By putting the lend lease event before, I am causing a 1 turn delay to set up lend lease.

Here, the Vichy card is given to Germany when Paris is occupied.

```

0) CHECK: Gameslot_Fall of France(#9) == 0
1) CHECK: CheckHexOwner(24, 23) == 0
2) SETVAR: Gameslot_Fall of France(#9) = 1
3) EXECUTE: ExecGiveActionCard(33, 0)
4) EXECUTE: ExecMessage2(0, -1, -1, -1)
5) END CHECK
6) END CHECK

```

The Fall of France triggers the Soviet Union to wake up. The Soviet DipBlock is removed so Germany may now declare war.

```

0) CHECK: Gameslot_Fall of France(#9) == 1
1) CHECK: CheckSleep(2) == 1
2) EXECUTE: ExecMessage2(0, 1, -1, -1)
3) EXECUTE: ExecSetSleep(2, 0)
4) EXECUTE: ExecChangeDipBlock(2)
5) END CHECK
6) END CHECK

```

The following events deal with some of the political aspects of the war.

If Germany does not declare war by 1942, Stalin renounces the non-aggression pact and declares war on Germany.

```

0) CHECK: CheckYear > 1941
1) CHECK: CheckTurn == 2
2) CHECK: CheckWar(0, 2) == 0
3) EXECUTE: ExecChangeDip(0, 2)
4) EXECUTE: ExecMessage2(0, 2, -1, -1)
5) END CHECK
6) END CHECK
7) END CHECK

```

Italy (regime 6) also joins the war and is folded into Germany with the Fall of France. Germany also gets the Italian PP bonus. *This was later changed to the fall of Lille*

```

0) CHECK: Gameslot_Fall of France(#9) == 1
1) CHECK: Gameslot_Italy in War(#7) == 0
2) EXECUTE: ExecJoinRegime(6, 0, 1)
3) SETVAR: Gameslot_Italy in War(#7) = 10
4) EXECUTE: ExecMessage2(0, 1, -1, -1)
5) END CHECK
6) END CHECK

```

Poland is surprised when attacked and suffers a readiness penalty. I could have set the readiness of the units at the start but then this gets reset if I ever hit the 'all units ready' button. Best to implement this as an event.

```

0) EXECUTE: ExecSetSleep(9, 0)
1) EXECUTE: ExecReduceReadiness(9, -1, 50, 100)
2) EXECUTE: BlockEvent

```

Next 2 events trigger Spain and Turkey. They join the other side if attacked. I show Spain below. Turkey is similar. The looper is used to role through the 3 major regimes.

```
0) LOOPER: TempVar0 FROM 0 TO 2
1) CHECK: CheckWar(TempVar0, 3) == 1
2) CHECK: CheckSleep(3) == 1
3) EXECUTE: ExecSetSleep(3, 0)
4) CHECK: TempVar0 == 0
5) EXECUTE: ExecJoinRegime(3, 2, 1)
6) END CHECK
7) CHECK: TempVar0 > 0
8) EXECUTE: ExecJoinRegime(3, 0, 1)
9) END CHECK
10) END CHECK
11) END CHECK
12) END LOOPER
```

Next set of events deal with victory and defeat. The first event checks for German Victory. I identified German Victory cities on the map (slot 1,1). In the first part of the event, I run a map check that counts the number of German Victory (e.g., Warsaw, Moscow) cities. If they have all 10, they win. *This condition has been tweaked*

```
0) SETVAR: Gameslot_German Victory Cities(#37) = 0
1) LOOPER: TempVar0 FROM 0 TO CheckMapWidth
2) LOOPER: TempVar1 FROM 0 TO CheckMapHeight
3) CHECK: CheckSlot(TempVar0, TempVar1, 1) == 1
4) CHECK: CheckHexOwner(TempVar0, TempVar1) == 0
5) SETVAR: Gameslot_German Victory Cities(#37) + 1
6) END CHECK
7) END CHECK
8) END LOOPER
9) END LOOPER
10) CHECK: Gameslot_German Victory Cities(#37) == 10
11) EXECUTE: ExecMessage2(0, 1, -1, -1)
12) EXECUTE: ExecSetWinner(0)
13) END CHECK
```

The next event executes the 'Establish Vichy' action card that the German player gets when France falls. I identified the Vichy areas with slot 2 and give the territories to Germany (in the case of Northern France) or the Vichy government. I then give the German player the 'Occupy Vichy' card and put a small garrison in Vichy controlled territories. Note that all Allied units in Northern France surrender.

```
0) EXECUTE: ExecSetSleep(11, 0)
1) EXECUTE: ExecJoinArea(2, 3, 0, -1)
2) EXECUTE: ExecJoinArea(2, 2, 11, -1)
3) EXECUTE: ExecJoinArea(2, 4, 11, -1)
4) EXECUTE: ExecJoinArea(2, 5, 11, -1)
5) EXECUTE: ExecGiveActionCard(34, 0)
6) EXECUTE: ExecMessage2(0, 1, -1, -1)
7) EXECUTE: ExecSetSleep(11, 1)
8) EXECUTE: ExecAddUnit(0, 20, 46, 11)
9) EXECUTE: ExecAddUnit(0, 3, 46, 11)
10) EXECUTE: ExecAddUnit(0, 26, 34, 11)
11) EXECUTE: ExecAddUnit(0, 74, 52, 11)
```

This next event executes the 'Occupy Vichy' action card. Germany now controls Southern France. The probability of partisan activity increases.

```
0) EXECUTE: ExecJoinArea(3, 2, 0, -1)
1) SETVAR: Gameslot_French Partisans(#24) + 1
2) EXECUTE: ExecMessage2(0, 1, -1, -1)
```

The event handles the West's declaration of War against Vichy (e.g., Operation Torch). First, the German 'Occupy Vichy' card is removed and Germany gets Southern France without the rise in Partisan level. The checks with Tempvar0 are an 'if – then – else'. There is a 25% chance that Vichy remains loyal to Germany. Otherwise, it joins the West. If Germany declares war on Vichy, it automatically joins the west. Note how I use BlockEvent once the event is triggered.

```
0) CHECK: CheckWar(1, 11) == 1
1) CHECK: CheckHexOwner(26, 34) > 0
2) EXECUTE: ExecRemoveActionCard(34, 0)
3) EXECUTE: ExecJoinArea(3, 2, 0, -1)
4) END CHECK
5) SETVAR: TempVar0 = 0
6) CHECK: CheckRandomPercent > 75
7) EXECUTE: ExecJoinRegime(11, 0, 1)
8) SETVAR: TempVar0 = 1
9) EXECUTE: ExecMessage2(0, 1, -1, -1)
```

```
10) END CHECK
11) CHECK: TempVar0 == 0
12) EXECUTE: ExecJoinRegime(11, 1, 1)
13) EXECUTE: ExecMessage2(0, 1, -1, -1)
14) END CHECK
15) EXECUTE: BlockEvent
16) END CHECK
17) CHECK: CheckWar(0, 11) == 1
18) EXECUTE: ExecJoinRegime(11, 1, 1)
19) EXECUTE: ExecMessage2(0, 1, -1, -1)
20) EXECUTE: BlockEvent
21) END CHECK
```

The above shows about a quarter of all the events used in the scenario. In larger scenarios, good organization becomes important. Within settings - group names, you can define event categories. This lets the designer group events into logical categories and makes for easier trouble shooting.