

Introversion Software DEFCON AI API

Quickstart and Documentation

Robin Baumgarten

November 10, 2008

Contents

1 Quickstart	2
2 Information on the DEFCON world	2
2.1 World Map	2
2.2 Game Time	3
2.3 Information Levels and Fog of War	3
2.4 Overview of the API Functions	3
2.5 Handling Events	3
2.6 Debugging	4
3 Setup and Installation	4
3.1 Installation of the Bot-Enabled Version of <i>DEFCON</i>	4
3.2 Installing Bots	4
4 Starting bots, hosting bot games and joining bot games	5
4.1 Play against Bots on other Clients	5
4.2 Playing against Bots on a Local Game	5
4.3 Avoiding Duplicate Key Messages	5
5 Bot Development	5
5.1 Command-line Options	5
5.2 Setting up Visual Studio	6
5.3 Debugging the DLL in Release Mode	7
6 Description of DEFCON	7
6.1 Parties & Territories	8
6.2 Alliances	8
6.3 Units	9

6.4	Ground Installations	9
6.5	Naval Units	9
6.6	Aerial Units	11
6.7	Temporal course of a DEFCON game	12
6.8	Winning conditions	12

1 Quickstart

This quickstart chapter will give you most information to get started. If you are not feeling like reading a lot of documentation, you can skip the remaining chapters for now and come back later should you get lost somewhere.

You can program your DEFCON AI Player ("*bot*" from now on) in at least three different programming languages: *C++*, *Python* and *Lua*. Minimal implementations and examples are provided in accordingly named subfolders of DEFCON\AI.

DEFCON will recognize all subdirectories in the DEFCON\AI directory as bots if they contain valid DLLs. These bots can be selected when you start a game or join a bot enabled game.

Also, there is a good and easy to use debugger included, which is deactivated by default and can be activated on the lobby screen or through commandline options. The provided examples show how to interact with it.

Defcon will call the bot every 100 ms[†]. Make sure that it does not take too long to run, or the game will hang!

2 Information on the DEFCON world

2.1 World Map

The game world is given by a cylindrical map where every object is represented by its longitude and its latitude, which can be thought of as *x* and *y* coordinates, respectively. The longitude ranges from -180.0 to 180.0 , the latitude ranges from -90.0 to 90.0 . As the map is cylindrical, the longitudes -180.0 and 180.0 represent the same longitude on the map. All longitudes received from and sent back to Defcon are normalised to be between the range given above.

Each coordinate of the map is either water or land. Naval units (*carriers*, *battle-ships* and *submarines*) can travel on most water bodies. Whether a coordinate is a water body can be through the `isValidTerritory(teamId, longitude, latitude, seaArea)` function. During the *Defcon 5* and *Defcon 4* states, units can be built on own territory. Land units (*airbases*, *radars* and *silos*) can only be built on land.

[†]Be aware that the amount of game time that passes in this 100 ms real time interval depends on the game speed!

2.2 Game Time

The server updates the world state every 100 ms. As the game speed is variable, the amount of time that passes in this time span varies by a factor of 0 (paused), 1 (real time), 5, 10, 20[‡]. The server calls the Update() function of each bot once during each world update.

2.3 Information Levels and Fog of War

It is possible to control the amount of information that the bots receive about the world. The selection is made when a new game is started:

Full information: When starting a game, all unit data is available to the bots, including positions, states, and targets.

Limited: This setting allows the bot to only see what a normal player would see.

2.4 Overview of the API Functions

The available functions can be classified into five categories which will be explained in the separate *defcon api reference.pdf* file:

- Functions concerning the game state
- Functions concerning teams (players)
- Functions concerning all units (getting information)
- Functions concerning own units (setting information)
- Debug functions

All provided functions only take and return simple data types (ints, floats, and strings) or vectors thereof. This leaves the choice of which programming paradigm to use up to the programmer. You are free to implement an object based structure that retrieves IDs from its object variables. In fact, the *C++* sample implementation shows one way how to use object orientation.

2.5 Handling Events

Events in *DEFCON* are handled through event messages, which will be passed to the *AddEvent* function in the API. Events are in-game actions such as launched nukes from subs and silos from other teams and requests such as other teams requesting to join the alliance, requests to share radar vision or a cease fire request. Events are also generated when the victory timer is started and the game is over. A list of the possible event types is given by the constants `EVENT_*`.

[‡]The current factor can be read out by `getGameSpeed()`

2.6 Debugging

There are a couple of in-game features that help you debug the game. To activate these tools, either select the debug option in the lobby window, or use the *debug* command line option. When activated, the following tools are available:

Timeline: On the bottom of the screen you'll find a timeline of the game. Events logged with *debugLog()* appear on the timeline as a vertical bar. When you click on the timeline, the game will replay the current scene, while the current running game is paused until you resume the game by navigating to the end of the timeline.

Log window: Events logged with *debugLog()* also appear in the log window. When you hover the mouse over a given entry, details of the entry are displayed in the information window.

Information window : This window is a more detailed version of the standard info window, and contains data of selected units or log entries, such as coordinates, unit ids and states.

3 Setup and Installation

This section will explain how to install the bot version of *DEFCON* and where to put bots to play against them.

3.1 Installation of the Bot-Enabled Version of *DEFCON*

You need this version to be able to host games with bots and to play on bot-enabled hosts with your own bot.

1. Download the *Windows* demo version of *DEFCON* here:
<http://www.introversion.co.uk/defcon/>
2. Download the bot-enabled executable *Defcon.exe* from here:
<http://www.introversion.co.uk/defcon/bots>
3. Install the demo and replace the demo *Defcon.exe* with the bot-enabled one

3.2 Installing Bots

Bots are installed by placing them in a subdirectory of *DEFCON\AI*, for example *C:\DEFCON\AI\myBot\bot.dll*. They are then available through a drop-down when hosting a game or joining a bot-enabled host.

4 Starting bots, hosting bot games and joining bot games

4.1 Play against Bots on other Clients

The bot-enabled version of *DEFCON* and the currently available "standard" version of *defcon* are *network compatible*, which means that you can play against a bot without downloading the bot-enabled version. To do that, either join a game with bots in them or host a game that allows for bots. All servers that potentially have bots in them have to have the prefix *[BOT]* or *[BOT-Debug]* in front of their names. If you launch or join a server named accordingly (e.g. *[BOT] My Server*), you can play against other bots, provided someone with a bot has joined the game.

4.2 Playing against Bots on a Local Game

When a bot is loaded into an instance of *DEFCON*, you cannot add an additional (human or bot) player on the same instance of the game. That means, if you want to play against a bot on one computer for example, you will have to start two instances of *Defcon*, one hosting the game and the other one joining in. It is possible to run several instances of *DEFCON* on the same computer. You may want to disable *Advertise on Internet* in the *Advanced Options* menu to avoid duplicate key messages, as explained in the next section.

4.3 Avoiding Duplicate Key Messages

Normally, when you join a game and another player in that game (or the host) has the same key as you, you are kicked from the server with a duplicate key error message. Because of the way the API implements bots, this is not desired: With this method you would have to have several keys if you want to play against two bots on one machine. To resolve this problem, duplicate key checks have been disabled in local area network games. To take advantage of this, disable *Advertise on Internet* in the *Advanced Options* when creating a new game.

5 Bot Development

5.1 Command-line Options

There are several command line options available to automate the start of the game and help debugging it. All option-names are case-insensitive, and values containing spaces should be wrapped in quotes. Format of command-line options: *optionname=value*, or *optionname*, or *optionname="values with spaces"*.

host : *DEFCON* automatically hosts a game. There are more command-line options that modify the game name, password etc, they are listed below.

join : *DEFCON* tries to connect to server specified by *servername* and *serverpassword*.

norender : In-game rendering is disabled.

demo : Starts the game with a demo key. Make sure to backup your original key somewhere, if you have one. It is located in the *authkey* file.

nolan : Disables advertising of server on the local area network.

nowan : Disables advertising of server on the internet (wide area network). This option disables duplicate key messages.

ai : Loads the specified bot when starting a game. Example: *ai="\AI\iv ai\iv ai.dll"*.

debug : Enables the debugging mode, with the timeline, logging window and info window.

limitedinformation : If set, bots will receive limited information only.

key : Uses specified authorization key.

playername : Sets the specified player name.

servername : When joining a game, *DEFCON* looks for the specified servername to join. This may also be an IP with port. Example: *servername="Testserver"* or *servername="123.12.1.23:5010"*. When creating a game, this will be the used server name.

password : Used password for either joining or creating a game.

numplayers : When hosting a game with *host*, the server will start the game (set itself to *Ready*) when the given number of players has joined (including the server itself).

5.2 Setting up Visual Studio

To be able to compile a dll for the use with *DEFCON* , follow these steps:

1. Obtain the code for an existing bot, for example simplebot. Place the code into any folder, for example *DEFCON\AI\myBot\source*.
2. Open Visual Studio and create a new project from the existing code. In our example, use *DEFCON\AI\myBot* as folder. If possible, select *Dynamic Link Library* as default project configuration type.
3. Set the configuration to Release.
4. The following settings will concern the project properties, viewable with Project->Properties. For the following settings, the location in the properties for Visual Studio 2008¹ have been given in brackets).

¹Both Visual Studio 2005 and 2008 have been used on this project, however it shouldn't matter which IDE you use.

5. (Optional) Set the output directory to $\$(SolutionDir)$ to create the .dll in the right place for *DEFCON* to find. (Configuration Properties - General)
6. Add *source/* as additional include directory (Configuration Properties - C/C++ - General)
7. Make sure Runtime Library is set to *Multi-threaded DLL (/MD)* (Configuration Properties - C/C++ - Code Generation)
8. Specify the Module Definition File to *source/dll/bot.def* (Configuration Properties - Linker - Input)

Now you should be able to compile a dll, which works with *DEFCON* .

5.3 Debugging the DLL in Release Mode

Although you have to compile the dll code in release mode (because the main program has been compiled in release mode), it is still possible to debug it:

1. Make sure you have debug information enabled. Set *Generate Debug Info* to *Yes (/DEBUG)* (Configuration Properties - Linker - Debugging).
2. Set up defcon.exe for debugging: Set the debug command to defcon.exe (e.g., *C:\DEFCON\defcon.exe*), and working directory to the directory containing *DEFCON* , e.g. *C:\DEFCON*. You can specify any command line arguments in *Command Arguments*, if you want. (Project Properties - Configuration Properties - Debugging)
3. When you start debugging (Debug - Start Debugging, or *F5*), the IDE will probably complain that defcon.exe was built without debug information. However, debugging still works, i.e., breakpoints should be hit and errors from the dll should show the code position.

6 Description of DEFCON

The following sections contain information about the Defcon units and the stages. If you are familiar with the game, you can skip these. Also, if you never played the game before, you should definitely play the demo² for a couple of games to get the hang of it.

DEFCON is a real-time strategy game, where the participating parties use offensive and defensive units to attack units and cities of opponents to score most points. The game is divided into stages, that sequentially unlock more powerful means of attack. The information in this chapter is derived from the manual³ and the game itself.

In this video game, each player plays a nuclear superpower, namely Europe, Africa, Russia, South Asia, North America or South America. At the beginning of a typical

²Demo available online at <http://www.everybody-dies.com/downloads/>.

³Available online at <http://www.everybody-dies.com/downloads/manual.html>.

match of *DEFCON*, every player chooses or is assigned one of these territories. The player then continues by placing structures like air defence silos, airbases and radar stations at tactical positions to protect his cities. Fleets are also created, they consist of battleships, carriers and submarines.

Thereafter, he starts commanding his units through the world map, in order to defend against attacks and start own attacks against the opponent. At first, fights ensue between fleets and planes. Players try to command their units for optimal efficiency, which includes selecting targets, form up fleets and choose between different unit statuses.

Finally, missiles are being used. They are devastating against structures and are the only way to score points, namely by striking opponent cities. The correct timing is important when a player plans to attack with missiles, concurrent attacks are more effective than dragged on attacks. Also opponent silos are defenceless when launching nukes themselves, which is a good opportunity to start a counter attack.

The game is finishes when most of the missiles have been used, the player who has inflicted the most damage to opponent cities wins.

The remainder of this chapter contains a formal description of *DEFCON* and its units, followed by a discussion of the AI bot created by Introversion.

6.1 Parties & Territories

A game has at least two parties, where each party can be controlled by either a human or an AI player. Each party controls one territory, which is randomly assigned or chosen before the game starts. Each territory can be controlled by 0 or 1 party. In each of the latter there is at least one city, each with a specified positive number of inhabitants in them. The sum of the inhabitants of all cities is equal for each territory. There are no cities in territories that are not controlled by a party. Also, a part of the sea (terrain in which naval units operate, disjunct from territories defined above) is available for each party to place naval units into in the first stages of the game course.

6.2 Alliances

Parties can form alliances to cooperate. This is done by sending a request to join the other party's alliance. All parties that are currently in this alliance can then vote whether or not to accept this request. If the majority of the voters accept the new party, it is added. The benefits of being in an alliance depend on the options set by the game server⁴. Ceasefire prevents units from attacking allied units, shared radar coverage enables parties to see all units of its allies, with the exception of submerged submarines. These settings can be changed while the game is running⁵. Allies can be expelled from an alliance through a majority vote, that has to be initiated by an alliance member. Allies themselves can leave an alliance at any time.

⁴Both automatic ceasefire and shared radar coverage are enabled by default.

⁵Only if the server settings allow for this (default).

6.3 Units

Each party has the same predefined quantity of units it can use. There are *ground installations*, *naval units* and *aerial units*.

6.4 Ground Installations

Silo: Initially contains 10 missiles.

States:

Air Defence: Automatically targets an aerial unit in attack range. If there are several, it chooses Nukes over Bombers over Fighters. If there are several of the same class, it chooses the closest. Targets can be manually assigned. After a shot is fired, it needs to recharge 20 seconds.

Launch Missiles: Targets for missiles have to be manually assigned. Every location on the map can be targeted. After a target is chosen, a missile will be launched. There is a recharge time of 120 seconds after the launch of a missile, whilst no other missiles can be launched. A launch of a missile will reveal the location of the Silo to all parties.

Radar: All Units (except submarines) within a certain distance are visible⁶ to the party. There are no actions for this ground installation.

Airbase: Initially contains 10 missiles, 5 fighters and 5 bombers. If there are fewer than the initial number of fighters, they are slowly regenerated, 1 fighter every 1000 seconds. The sum of the number of fighters and bombers contained in the airbase cannot exceed the initial combined size of 10 planes.

States:

Launch Fighters: Targets for fighters have to be manually assigned. Every location on the map can be targeted. After a target is chosen, a fighter will be launched. The recharge time after the launch of a fighter, whilst no other fighters can be launched is 20 seconds.

Launch Bombers: Same actions as in state *launch fighters*, with the difference that bombers will be launched instead of fighters.

6.5 Naval Units

All naval units can move to every reachable position on sea. Groups of up to 6 naval units can be combined to a *fleet* when they are initially placed. Thus a player usually has several fleets. A fleet cannot be separated and ships in fleets cannot be navigated separately, however it is possible to command ships in fleets to attack different targets and change into different states.

⁶ *To be visible* means *within radar range of any unit of the party* in this report.

Carrier: Initially contains 6 missiles, 5 fighters and 2 bombers. Maximum capacity equals initial number of fighters and bombers.

States:

Launch Fighters: If there are hostile units that can be attacked by fighters within a certain range of the carrier, the carrier will launch fighters. Targets for fighters can also be manually assigned. Every location on the map can be targeted. After a target is chosen, a fighter will be launched. There is a recharge time of 120 seconds after the launch of a fighter, whilst no other fighters can be launched.

Launch Bombers: Targets for bombers can also be manually assigned. Every location on the map can be targeted. After a target is chosen, a bomber will be launched. There is a recharge time of 120 seconds after the launch of a bomber, whilst no other bombers can be launched.

Anti Submarine: A sonar scan reveals all submarines within a certain distance from the carrier. If there are hostile submarines detected, a depth charge is released, which destroys hostile submarines within a certain range and probability. Each sonar scan has a recharge time of 60 seconds.

Battleship: Automatically attacks hostile naval units, fighters and bombers within its attack range.

Submarine: Initially contains 5 missiles.

States:

Passive Sonar: If not submerged already, a submarine does so upon entering this state. It is then invisible to radar and can only be detected by carriers in anti submarine status and other submarines in active sonar mode. It can attack hostile naval units if they are visible to the party of the submarine.

Active Sonar: The same as passive sonar, additionally the submarine creates a sonar scan, which reveals all naval units within a certain distance from the submarine. Each sonar scan has a recharge time of 20 seconds.

Launch Missiles: Submarine surfaces upon entering this state, thus becoming visible to radar and attackable by battleships, submarines, fighters and bombers. Targets for missiles have to be manually assigned. Every location within a certain attack range can be targeted. After a target is chosen, a missile will be launched. There is a recharge time of 120 seconds after the launch of a missile, whilst no other missiles can be launched.

A launch of a missile will reveal the location of the submarine to all parties. Once all missiles have been launched, the submarine enters passive sonar state. The launch missiles state can only be chosen if the submarine still contains missiles.

6.6 Aerial Units

Fighter: Attacks visible hostile surfaced naval units and fighters and bombers within its attack range. A fighter has limited fuel. Any unit with limited fuel will crash if it does not return to an airbase or carrier before it runs out of fuel.

Bomber: Like the fighter, the bomber has limited fuel.

States:

Naval Combat: Attacks visible hostile surfaced naval units within its attack range.

Launch Missiles: Targets for missiles have to be manually assigned. Every location can be targeted. When the target is within the bombers attack range, the missile will be launched. Otherwise the targets can be reassigned.

Missile: A missile can be disarmed. This action destroys the missile after 100 seconds, if it has not impacted before. Upon impact, a missile inflicts damage to all units within a radius of 0.5 length units. (see table 1).

Structure	Hits to destroy
Radar	1
Airbase	2
Silo	3

Table 1: Hits of missiles required to destroy structures

State	Time [sec]
Silo: Air Defence	340
Silo: Launch Missiles	120
Airbase: Launch Fighters	120
Airbase: Launch Bombers	120
Carrier: Launch Fighters	120
Carrier: Launch Bombers	120
Carrier: Anti Submarine	240
Submarine: Passive Sonar	240
Submarine: Active Sonar	240
Submarine: Launch Missiles	120
Bomber: Naval Combat	60
Bomber: Launch Missiles	240

Table 2: Default time to change into given state

Attacker	Target							
	Fighter	Bomber	Battleship	Carrier	surfaced Sub	submerged Sub	Missile	
Fighter	medium	very high	medium	medium	high	-	-	
Bomber	-	-	high	high	very high	-	-	
Battleship	very high	high	medium	high	very high	-	-	
Carrier	-	-	-	-	very high	very high	-	
surfaced Sub	-	-	-	-	-	-	-	
submerged Sub	-	-	high	high	high	high	-	
Silo	low	medium	-	-	-	-	very high	

Table 3: Chances of destruction of target upon hit. The probabilities convert as follows:
low = 10%, medium = 20%, high = 25%, very high = 30%

6.7 Temporal course of a DEFCON game

A game is divided into 4 stages, each called a defcon phase by the game. It starts in stage Defcon 5.

Defcon 5: A party can place⁷ its ground installations within its territories and naval units within its assigned sea parts. It cannot see nor attack any unit of other parties, even if they would be in radar range. Naval units can move within waters that are not assigned to other parties. Other actions are not possible. This stage lasts 3 Minutes.

Defcon 4: The same as Defcon 5, except that units of other parties can now be seen, if in radar range. This stage lasts three minutes.

Defcon 3: In this stages units can no longer be placed. All actions not involving missiles are allowed, i.e. non-missile aerial units can be launched, naval units can move into foreign waters, attacks are possible. This stage lasts 3 minutes.

Defcon 2: In this stage, the same conditions as in Defcon 3 apply. It lasts 3 minutes.

Defcon 1: Units cannot be placed. All actions are allowed. This stage has no fixed time limit. After 80% of all missiles in the game have been launched, a *victory counter* starts (45 Minutes). When this counter finishes, the game ends.

6.8 Winning conditions

The score of a party in *DEFCON* is calculated as $s = a\Delta k - b\Delta l$ where k denotes the number of points scored in opponent cities and l denotes the number of points scored by opponents in their own cities. a and b are defined by the score mode, that has to be chosen before the game starts (see table 4). A missile that hits a city increases the score by 50% of the inhabitants in that city (measured in millions) and decreases the number of inhabitants by 50%. At default score mode, a party therefore scores 2 points for each million inhabitants that opponent cities were reduced by through own missiles, and gets -1 point for each million of inhabitants that were lost in own cities. At the end of the game the party with the highest score wins.

⁷Placing a unit is the action of selecting it out of a pool (which is outside the world) of available units

Score mode	a	b
Default	2	1
Genocide	1	0
Survivor	0	1

Table 4: score modes in *DEFCON*

and placing them in the world.