DEFCON API Function List

Group	Name	Arguments	type	Returns / Description
World state	**GetDefcon**		int	Current Defcon Stage, game starts with 5
	GetGameTime		float	Current Game Time, measured in seconds. Each tick, the game progresses by 0.1 sec * GameSpeed
	GetGameTick		int	Amount of update cycles (ticks) passed since game start
	GetGameSpeed		int	Current speed-up factor of the game over the real time passed. Usually has values from 0 (paused), 1 (real time), 5, 10, 20, see enum GAMESPEED_*
	GetVictoryTimer		float	Time remaining in game, if victory timer was started. Test this with IsVictoryTimerStarted
	IsVictoryTimerActive		bool	True iff the victory-timer has been started
	GetOptionValue	char *	int	Value of certain option
Cities	**GetCityIds**		array (int)	Array of City Ids. The amount of cities does not change during a game
	GetCityPopulation	CityId	int	Population (in millions)
	GetRemainingPopulation	TeamId	int	Remaining population of given team
Worldmap	**IsValidTerritory**	teamId, longitude, latitude, seaArea	bool	True if the given coordinates belong to the given Team. If seaArea is set to true, then Coordinates must be on sea area, otherwise land. If teamId = -1, then function returns if coordinates are land or sea terrain respectively. Note that there can be coordinates which are neither land nor sea
	IsBorder	longitude, latitude	bool	True if given coordinates are on the border. Compare "data/earth/coastlines.bmp"
	GetTerritoryId	longitude, latitude	int	Territory Id of territory at given coordinates
Teams	**GetOwnTeamId**		int	Own team id
	GetTeamIds		array (int)	List of Team Ids in the game
	GetTeamTerritoryCount	teamId	int	Number of territories for given team, usually 1
	GetTeamTerritories	teamId	array (int)	Territory Ids of territories that the given team owns. The enum TERRITORY_* relates the ids to starting positions
	GetAllianceId	teamId	int	Id of alliance. Each team belongs to exactly one alliance
	GetDesiredGameSpeed	teamId	int (enum)	Currently requested game speed of given team
Scores	**GetEnemyKills**	teamId	int	Sum of enemy kills of the given team (for scoring)
	GetFriendlyDeaths	teamId	int	Sum of friendly deaths (deaths in allied populations) of the given team
	GetCollateralDamage	teamId	int	Sum of collateral damage deaths (deaths in own population) of the given team
	GetTeamName	teamId	String	Name of the given team
	IsSharingRadar	teamId, teamId	bool	True iff the first team is sharing its radar with the second team
	IsCeaseFire	teamId, teamId	bool	True iff the first team is in cease fire mode with the second team
Alliance	**RequestAlliance**	allianceId		Sends requests to the alliance members to join alliance. Replies are handled by the event system
	RequestCeaseFire	teamId		Send request to cease fire with given team
	RequestShareRadar	teamId		Send request to share radar with given team
	RequestGameSpeed		int	Send request to change game speed to given speed. Must be one of the values specified in GAMESPEED_*
Lists	**GetAllUnits**		array (int)	All visible unit ids
	GetAllOwnUnits		array (int)	All own unit ids
	GetTeamUnits	teamId	array (int)	All visible units of a given team
	GetAllUnitData		array (unitData)	Data about all visible units, contained in the struct unitData (see enums). This function is for convenience only, as all data can be gathered through other functions, too
	GetType	unitId, eventId or teamId	int (enum)	Type of unit, event or team, specified in enum TYPE_*, EVENT_* or TEAM_TYPE_*
	GetTeamId	unitId or eventId	int	Team Id of given unit
Fleets	**GetOwnFleets**		array (int)	Own fleet ids
	GetFleets	teamId	array (int)	Fleet ids of given team. Only fleets ids with visible members are returned
	GetFleetMembers	fleetId	array (int)	Ids of ships in given fleet
	GetFleetId	unitId	int	Id of fleet of given unit
Gunfire/Depthch.	**GetShots**		array (int)	Ids of all visible shots (projectiles like gunfire and depth charges)
	GetShotOrigin	shotId	int	UnitId of originator of given shot, if visible at shooting time
	GetShotOriginLocation	shotId	array(int)	Location where shot has been first seen
Unit states	**GetCurrentState**	unitId	int (enum)	State of unit, specified in enum STATE_*
	GetCurrentStateCount	unitId	int	Number of activations of current state in given unit, i.e., number of nukes in a sub or silo, number of planes available in a carrier or airbase
	GetStateCount	stateId, unitId	int	Number of activations of given state in given unit
	GetStateTimer	unitId	float	Time until current state is active
	GetActionQueue	unitId	array (int)	Array of unitIds of currently queued actions, for example nukes in a silo or planes on a carrier
	GetCurrentTargetId	unitId	int	Current target id. -1 if no target is set or target is location. If track of target is lost, the last known location is used instead
	GetMovementTargetLocation	unitId	array (float)	Current target location. (0,0) if no target location is set
	GetNukeSupply	unitId	int	Number of available nukes
	GetBomberNukeTarget	unitId	array (float)	Target of the nuke carried by given bomber
	IsRetaliating	unitId		True iff given unit is automatically retaliating an earlier attack

	IsVisible	unitId, byTeamId	bool	True iff given unit is visible to given team. In full information mode, visibility information about other teams is available. In limited information mode, only visible units are accessible.
	SetState	unitId, StateId		Set state of given unit. See STATE_*
	SetTargetLocation	unitId, longitude, latitude		Set target location for given unit. If target id is also given, the id overrides the location
	SetTargetId	unitId, targetUnitId		Set target unit id for given unit
Movement	**GetLongitude**	unitId or cityId or eventId	float	Longitude of given unit, city, or event
	GetLatitude	unitId or cityId or eventId	float	Latitude of given unit, city, or event
	GetVelocity	unitId	array (float)	Movement direction of given unit, in longitude and latitude parts. The vector has the length of the unit speed (see also SPEED_*)
	GetRange	unitId	float	Remaining range of unit. If unlimited, -1 is returned
Setup	**GetRemainingUnits**	typeId	int	Amount of remaining units of given type that can be placed
	IsValidPlacementLocation	longitude, latitude, typeId	bool	True iff given location is valid for placement of given type. For fleets use getFleetMemberOffset to get offset from fleet center
	GetFleetMemberOffset	memberCount, memberId	vector<float>	Offset of ship number memberId from center of fleet, given fleet has memberCount ships
	PlaceStructure	typeId, longitude, latitude		Tries to place a given structure to the given coordinates. Use IsValidStructureLocation to test if valid
	PlaceFleet	longitude, latitude, shipType1Id, ..., shipType6Id		Tries to place a given amount of battlecruisers, carriers and subs into a fleet at the given location. Use IsValidFleedLocation to test
	GetUnitCredits		int	Credits available for placement (if in variable unit mode)
	GetUnitValue	typeId	int	Value of given unit type (if in variable unit mode)
Events	**SendEventAgree**	eventId		Agrees to a given event that can be agreed on, eg. alliance requests, cease fire requests, etc
	SendEventDeny	eventId		Denies a given event that can be denied, eg. alliance requests, cease fire requests, etc
	SendChatMessage	string		Sends a chat message
Tools/Geometry	**GetDistance**	longitude1, latitude1, longitude2, latitude2	float	Distance in game between given coordinates
	GetSailDistance	longitude1, latitude1, longitude2, latitude2	float	Distance in game between given coordinates on sea (performs pathfinding)
	GetSuccessfulCommands		array (int)	CommandIds of all commands that have been executed in previous cycle
Debug	**DebugLog**	String, unitId, tags, R,G,B, alpha		Prints a line in the debug console in the specified color
	DebugIsReplayingGame		bool	True if the game is currently replayed (Timeline has been clicked)
	DebugWhiteboardDraw	longitude1, latitude1, longitude2, latitude2		Draws a line on the whiteboard
	DebugWhiteboardClear			Clears the whiteboard
	DebugMoveCamera	longitude, latitude, zoom		Moves the camera to a given location