# Specification of Requirements for a Critter Controller

## Contents

## The Visual Studio Solution

You have been provided with a Visual Studio 2017 solution called Critterworld. The solution, as provided, will build and execute so you can see the actions of the two demonstration Critters. You can use the demonstration Critter controllers as the starting point for your own implementations.

## Writing a Controller for a Critter

When Critterworld executes, it looks for dynamic link library files (DLLs) in the same folder as itself, or a folder specified in its configuration (accessible via the main menu[1]). In each DLL it finds, Critterworld looks for a class that implements the interface ICritterControllerFactory. If a class that implements ICritterControllerFactory is found, a new instance of that class is created.

The DLL must also contain classes that define your Critter controllers.

*To create one or more working Critters, you need to write a class that implements ICritterControllerFactory (the loader class) and one or more classes that implement ICritterController (a Critter controller class).*

## The Loader Class

The DLL must include a class that creates one or new controllers and provides a way for Critterworld to load them into memory. To do this, you must write a class that implements the interface ICritterControllerFactory.

The following is a definition of ICritterControllerFactory:

```
public interface ICritterControllerFactory
{
    string Author { get; }
    ICritterController[] GetCritterControllers();
}
```

It consists of one property and one method:

| | |
|---|---|
| Author | This property must return the name and student number of the author of the Critter controllers, i.e. you. It must be in the form *first_name last_name student_number*, e.g.,<br>Dave Voorhis 123456789<br>**It must not contain any control characters (e.g., tab or newline) or colons (:).** |
| GetCritterControllers | This method must return an array of instances of ICritterController. |

Each ICritterController instance is a controller you have implemented. This approach allows the DLL to contain more than one controller, as is done in DemonstrationCritters.dll which provides two demonstration critters. It also makes it possible to return multiple controller instances of a given controller type. For example, assuming you have created three different

[1] As of this writing, not yet available, but to be available shortly.

controllers, *Fighter*, *Seeker* and *Hunter*, you might have GetCritterControllers() return one instance of Fighter, two instances of Seeker, and two instances of Hunter.

## *The Controllers*

A Critter controller is conceptually simple: It receives messages from the Critterworld system containing information about its Critter (and associated environment) and sends messages to the Critterworld system to control its Critter.

The messages the controller receives indicate what the Critter can sense at long-range with its bug-like antennae, what the Critter can see nearby with its rather near-sighted eyes, how much time a level runs for, how much time remains for the level to run, when a level starts or finishes, the dimensions of the current level, the current position of the Critter, and its current speed (in a direction), and other things.

The controller can send messages to tell the Critter to move at a given speed in a particular direction, to move at a random speed in a random direction, to use its antennae to sense long-range objects, to ask how much time a level runs for or how much time remains for a level to run, to stop moving, to request the dimensions of the current level, request the current position and/or speed of the Critter, and so on.

To create a controller, you must write a class that implements the ICritterController interface. You mainly define behaviour for your critter by implementing the methods specified by ICritterController to send and receive messages from the Critterworld environment.

ICritterController is specified as follows:

```
public delegate void Send(string message);

public interface ICritterController
{
    string Name { get; }
    Send Responder { get; set; }
    Send Logger { get; set; }
    string Filepath { get; set; }
    void Receive(string message);
    void LaunchUI();
}
```

The *Send* delegate is not part of the interface but is referenced by it and so is included here.

The properties and methods are described below:

Name — This property returns the name of the Critter. This is the name displayed in the various tables in the Critterworld user interface. **You may choose any name you like, but it should be shorter than 15 characters and must not be anything vulgar, political, offensive, or inappropriately suggestive. Critterworld may be run publicly, so inappropriate critter names will result in the submission being removed and its creator will receive a failing grade.**

**It must not contain any control characters (e.g., tab or newline) or colons (:).**

If *null* is returned, Critterworld will automatically generate a random name.

Responder This property specifies the method that is invoked to send a message from the controller to the Critterworld environment to make inquiries or control the Critter. It is normally invoked via…

```
Responder(message);
```

…where *message* is a string. The message format is specified in its own section below.

This property is set by the Critterworld environment and must not be changed by the controller.

Logger This property specifies the method that is invoked to log error or status messages generated by the Critter controller when the *debug* checkbox is selected on a Critter's scoreboard (to the right of the arena) in the Critterworld window.

It is normally invoked via…

```
Logger(message);
```

…where *message* is any string. However, the strings should be kept sort and sent infrequently to avoid causing the Critterworld environment to become slow and unresponsive.

This property is set by the Critterworld environment and should not be changed by the controller.

Filepath This property specifies the file path where files specific to this Critter controller may be stored.

This property is set by the Critterworld environment and must not be changed by the controller.

Any files created by the Controller must be stored in the directory specified by this path and nowhere else. **Reading or writing files anywhere else is strictly forbidden and may incur serious penalties!**

Receive This method is invoked by the Critterworld environment whenever the Critterworld environment needs to send a message to the controller. This method will be invoked frequently and therefore must spend as little time as possible processing each message, to prevent the Critterworld environment from becoming slow and unresponsive.

Lengthy operations should be handled in their own threads, and such threads must be defined with their IsBackground property set to true or otherwise shut down when a Critter controller receives a Shutdown message.

The messages that may be processed by the Receive method are detailed in their own section below.

LaunchUI This method is invoked by the Critterworld environment whenever the Critter "settings" icon (a meshed pair of gears) is clicked on a Critter's scoreboard. It must open a dialog box to allow the user to, for example…

- Alter configuration settings specific to the Critter controller;
- View status information about the running Critter;

- View maps or other data maintained by the Critter;
- Etc.

…or close such a dialog box if it is already open.

The first item – alter configuration settings specific to the Critter controller – must be implemented. The others are optional, but almost certainly useful during development and debugging.

## *Messages Sent to the Critterworld Environment*

This section documents the messages, and their format, that can be sent to the Critterworld environment via *Responder(message)*, as described above.

Note: In the following, any item specified between angle brackets <…> represents a literal value. The angle brackets and identifier name are part of the specification; they do not appear in the data. E.g., a valid <request-number> is 34.

Note: When shown below, the <request-number> is arbitrary and provided by the user. It will be returned in the associated response. It may be used to match requests with responses when multiple requests are sent, as responses are not guaranteed to be received in any order, or necessarily at all.

| | |
|---|---|
| SCAN | Request the controlled Critter to scan its environment using its antennae. The response is a SCAN message from the Critterworld environment, described below. |
| Message format: | SCAN:<request-number> |
| Example: | Responder("SCAN:34"); |

| | |
|---|---|
| GET_LEVEL_DURATION | Request the duration of this level in seconds. The response is a LEVEL_DURATION message from the Critterworld environment, described below. |
| Message format: | GET_LEVEL_DURATION:<request-number> |
| Example: | Responder("GET_LEVEL_DURATION:34"); |

| | |
|---|---|
| GET_LEVEL_TIME_REMAINING | Request the time remaining in this level in seconds. The response is a LEVEL_TIME_REMAINING message from the Critterworld environment, described below. |
| Message format: | GET_LEVEL_TIMER_REMAINING:<request-number> |
| Example: | Responder("GET_LEVEL_TIME_REMAINING:34"); |

| | |
|---|---|
| GET_HEALTH | Request the health level of the Critter. The response is a HEALTH message from the Critterworld environment, described below. |
| Message format: | GET_HEALTH:<request-number> |
| Example: | Responder("GET_HEALTH:34"); |

| | |
|---|---|
| GET_ENERGY | Request the energy level of the Critter. The response is an ENERGY message from the Critterworld environment, described below. |
| Message format: | GET_HEALTH:<request-number> |
| Example: | Responder("GET_HEALTH:34"); |

| | |
|---|---|
| GET_LOCATION | Request the location of the Critter in the arena, based on pixel coordinates. The response is a LOCATION message from the Critterworld environment, described below. |
| Message format: | GET_LOCATION:<request-number> |
| Example: | Responder("GET_LOCATION:34"); |

| | |
|---|---|
| GET_SPEED | Request the speed of the Critter in the arena. The response is a SPEED message from the Critterworld environment, described below. |
| Message format: | GET_SPEED:<request-number> |
| Example: | Responder("GET_SPEED:34"); |

| | |
|---|---|
| GET_ARENA_SIZE | Request the size of the arena in pixels. The response is an ARENA_SIZE message from the Critterworld environment, described below. |
| Message format: | GET_ARENA_SIZE:<request-number> |
| Example: | Responder("GET_ARENA_SIZE:34"); |

| | |
|---|---|
| GET_ARENA_SIZE | Request the size of the arena in pixels. The response is an ARENA_SIZE message from the Critterworld environment, described below. |
| Message format: | GET_ARENA_SIZE:<request-number> |
| Example: | Responder("GET_ARENA_SIZE:34"); |

| | |
|---|---|
| SET_DESTINATION | Request that the Critter begin progressing to a specified destination at a specified speed.<br><br>The speed must be a value between 0 (stopped) and 10 (full speed).<br><br>The x-coord and y-coord coordinates must lie within the arena. |
| Message format: | SET_DESTINATION:<x-coord>:<y-coord>:<speed> |
| Example: | Responder("SET_DESTINATION:50:75:2"); |

| | |
|---|---|
| RANDOM_DESTINATION | Request that the Critter begin progressing to a random destination at a random speed. |
| Message format: | RANDOM_DESTINATION |
| Example: | Responder("RANDOM_DESTINATION"); |

| | |
|---|---|
| STOP | Request that the Critter stop moving. |
| Message format: | STOP |
| Example: | Responder("STOP"); |

| | |
|---|---|
| SET_SPEED | Request that the Critter resume moving at a specified speed.<br><br>The speed must be a value between 0 (stopped) and 10 (full speed). |
| Message format: | SET_SPEED:<speed> |
| Example: | Responder("SET_SPEED:5"); |

## *Messages Received from the Critterworld Environment*

This section documents the messages, and their format, that can be received from the Critterworld environment via the *Receive* method, as described above.

Note: In the following, any item specified between angle brackets <…> represents a literal value. The angle brackets and identifier name are part of the specification; they do not appear in the data. E.g., a valid <request-number> is 34.

Note: In the following, \n represents a newline character; \t represents a tab character.

| | |
|---|---|
| ERROR | Indicates a problem, typically caused by an invalid request message to the Critterworld environment. |
| Message format: | ERROR: <error message> |
| Example: | ERROR: Unknown command: test |

| | |
|---|---|
| SEE | This message is repeatedly and automatically sent by the Critterworld environment to indicate what the Critter "sees" within a limited radius. It is roughly analogous to an insect's eyes. |
| Message format 1 – nothing nearby: | SEE: \n<br>Nothing |
| Message format 2 – objects nearby: | SEE: \n*One or more of any of the following separated by tab (\t) characters:*<br><br>Bomb:<coordinate><br>Critter:<coordinate>:<critter-number>:<name> by <author>:<health-status>:<alive-or-dead><br>EscapeHatch:<coordinate><br>Food:<coordinate><br>Gift:<coordinate><br>Terrain:<coordinate><br><br>Where:<br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena<br><critter-number> is a unique number identifying a Critter<br><name> is a Critter name |

| | |
|---|---|
| | <author> is a Critter author identification<br><health-status> is Strong, Ok, Adequate, Weak<br><alive-or-dead> is Alive or Dead |
| Example 1: | SEE:<br>Nothing |
| Example 2: | SEE:<br>Food:{X=222,Y=91}\tFood:{X=182,Y=129}\tFood:{X=223,Y=108}\tCritter:{X=276,Y=208}:3:Wanderer2 by Dave Voorhis:Strong:Alive |

| | |
|---|---|
| SCAN | This message is sent by the Critterworld environment in response to receipt of a SCAN request (see above). It is roughly analogous to an insect's antennae and provides information on the location of various near and distant objects. |
| Message format: | SCAN:<request-number>\n*Zero or more of any of the following separated by tab (\t) characters:*<br><br>EscapeHatch:<coordinate><br>Food:<coordinate><br>Gift:<coordinate><br><br>Where:<br><request-number> is the user-supplied request number<br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | SCAN:34<br>Food:{X=364,Y=782}\tFood:{X=639,Y=528}\tFood:{X=720,Y=161}\tFood:{X=518,Y=694}\tFood:{X=895,Y=155}\tFood:{X=659,Y=162}\tFood:{X=971,Y=546}\tFood:{X=900,Y=196}\tFood:{X=694,Y=122}\tFood:{X=198,Y=25}\tFood:{X=1069,Y=533}\tFood:{X=640,Y=634}\tFood:{X=279,Y=788}\tFood:{X=888,Y=180}\tFood:{X=296,Y=523}\tFood:{X=281,Y=746}\tFood:{X=693,Y=705}\tFood:{X=999,Y=68}\tFood:{X=397,Y=347}\tFood:{X=1020,Y=560}\tGift:{X=417,Y=311}\tGift:{X=860,Y=768}\tGift:{X=750,Y=333}\tGift:{X=554,Y=282}\tGift:{X=263,Y=495}\tGift:{X=263,Y=338}\tGift:{X=748,Y=792}\tGift:{X=910,Y=472}\tGift:{X=563,Y=603}\tGift:{X=623,Y=518}\tGift:{X=508,Y=171}\tGift:{X=801,Y=145}\tGift:{X=179,Y=507}\tGift:{X=705,Y=27}\tGift:{X=854,Y=604}\tGift:{X=1042,Y=151}\tGift:{X=330,Y=305}\tGift:{X=819,Y=504}\tGift:{X=977,Y=470}\tGift:{X=276,Y=600}\tEscapeHatch:{X=1031,Y=391} |

| | |
|---|---|
| LEVEL_DURATION | This message is sent in response to receipt of a GET_LEVEL_DURATION message (see above). It indicates the duration, in seconds, of the current level. |
| Message format: | LEVEL_DURATION:<request-number>:<seconds><br><br>Where: |

| | |
|---|---|
| | <request-number> is the user-supplied request number<br><seconds> is the number of seconds |
| Example: | LEVEL_DURATION:34:120 |

| | |
|---|---|
| LEVEL_TIME_REMAINING | This message is sent in response to receipt of a GET_TIME_REMAINING message (see above). It indicates the time remaining, in seconds, in the current level. |
| Message format: | LEVEL_TIME_REMAINING:<request-number>:<seconds><br><br>Where:<br><br><request-number> is the user-supplied request number<br><seconds> is the number of seconds |
| Example: | LEVEL_TIME_REMAINING:34:6 |

| | |
|---|---|
| HEALTH | This message is sent in response to receipt of a GET_HEALTH message (see above). It indicates the current health of the Critter. |
| Message format: | HEALTH:<request-number>:<value>:<health-status><br><br>Where:<br><br><request-number> is the user-supplied request number<br><value> is the health percentage, between 0 and 100<br><health-status> is Strong, Ok, Adequate, Weak<br><br>Strong is a health percentage > 75<br>Ok is a health percentage > 50 and ≤ 75<br>Adequate is a health percentage > 25 and ≤ 50<br>Weak is a health percentage ≤ 25 |
| Example: | HEALTH:34:78:Strong |

| | |
|---|---|
| ENERGY | This message is sent in response to receipt of a GET_ENERGY message (see above). It indicates the energy level of the Critter. |
| Message format: | ENERGY:<request-number>:<value><br><br>Where:<br><br><request-number> is the user-supplied request number<br><value> is the energy percentage, between 0 and 100 |
| Example: | ENERGY:34:22 |

| | |
|---|---|
| LOCATION | This message is sent in response to receipt of a GET_LOCATION message (see above). It indicates the location of the Critter. |
| Message format: | LOCATION:<request-number>:<coordinate> |

| | |
|---|---|
| | Where:<br><br><request-number> is the user-supplied request number<br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | LOCATION:34:{X=145,Y=352} |

| | |
|---|---|
| SPEED | This message is sent in response to receipt of a GET_SPEED message (see above). It indicates the velocity of the Critter. |
| Message format: | SPEED:<x-velocity>:<y-velocity>:<velocity><br><br>Where:<br><br><x-velocity> is horizontal velocity<br><y-velocity> is vertical velocity<br><velocity> is non-directional velocity |
| Example: | SPEED:34:2:1.036924:0.9630763 |

| | |
|---|---|
| ARENA_SIZE | This message is sent in response to receipt of a GET_ARENA_SIZE message (see above). It indicates the size of the arena in pixels. |
| Message format: | ARENA_SIZE:<request-number>:<horizontal_size>:<vertical_size><br><br>Where:<br><br><request-number> is the user-supplied request number<br><horizontal_size> is the horizontal extent of the arena<br><vertical_size> is the vertical extent of the arena |
| Example: | ARENA_SIZE:34:1020:800 |

| | |
|---|---|
| ESCAPE | This message is sent when a Critter exits through the escape hatch. The received coordinate is the position of the Critter at the point where it left the level.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received.** |
| Message format: | ESCAPE:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | ESCAPE:{X=145,Y=352} |

| | |
|---|---|
| SCORED | This message is sent when a Critter obtains a Gift. The received coordinate is the position of the Critter at the point where it received the Gift. |
| Message format: | SCORED:<coordinate>:<score><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena<br><score> is the Critter's score on this level |
| Example: | SCORED:{X=145,Y=352}:55 |

| | |
|---|---|
| ATE | This message is sent when a Critter eats a piece of Food. The received coordinate is the position of the Critter at the point where it ate the Food. |
| Message format: | ATE:<coordinate>:<energy>:<health><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena<br><energy> is the Critter's energy level after eating<br><health> is the Critter's health level after eating |
| Example: | ATE:{X=145,Y=352}:55:60 |

| | |
|---|---|
| FIGHT | This message is sent when a Critter bumps into another Critter. This is treated as fighting and will reduce both Critters' health. The received coordinate is the position of the Critter at the point where it bumped into another Critter. |
| Message format: | FIGHT:<coordinate>:<critter-number>:<name> by <author><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena<br><critter-number> is a unique number identifying a Critter<br><name> is a Critter name<br><author> is a Critter author identification |
| Example: | FIGHT:{X=145,Y=352}:3:Wanderer by Dave Voorhis |

| | |
|---|---|
| BUMP | This message is sent when a Critter bumps into Terrain. The received coordinate is the position of the Critter at the point where it bumped into Terrain. |
| Message format: | BUMP:<coordinate><br><br>Where: |

| | |
|---|---|
| | <coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | BUMP:{X=145,Y=352} |

| | |
|---|---|
| FATALITY | This message is sent when a Critter dies due to health declining to zero. The received coordinate is the position of the Critter at the point where its health declined to zero.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received.** |
| Message format: | FATALITY:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | FATALITY:{X=145,Y=352} |

| | |
|---|---|
| STARVED | This message is sent when a Critter dies due to energy declining to zero. The received coordinate is the position of the Critter at the point where its energy declined to zero.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received.** |
| Message format: | STARVED:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | STARVED:{X=145,Y=352} |

| | |
|---|---|
| BOMBED | This message is sent when a Critter dies due running into a bomb. The received coordinate is the position of the Critter at the point where it hit a bomb.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received.** |
| Message format: | BOMBED:<coordinate><br><br>Where: |

| | |
|---|---|
| | <coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | BOMBED:{X=145,Y=352} |

| | |
|---|---|
| CRASHED | This message is sent when a Critter dies due to an exception being thrown in its Controller code. The received coordinate is the position of the Critter at the point where it crashed.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received.** |
| Message format: | CRASHED:<coordinate>:<reason><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena<br><reason> a string with information about the crash |
| Example: | CRASHED:{X=132,Y=12}:System.NullReferenceException: Object reference not set to an instance of an object.<br>at DemonstrationCritters.Wanderer.Receive(String message) in C:\Users\voorh\source\repos\CritterWorld\DemonstrationCritters\Wanderer.cs:line 49<br>at CritterWorld.Critter.<Launch>b__134_1() in C:\Users\voorh\source\repos\CritterWorld\CritterWorld\Things\Critter.cs:line 645 |

| | |
|---|---|
| REACHED_DESTINATION | This message is sent when a Critter reaches a destination location set via a RANDOM_DESTINATION or SET_DESTINATION message. The received coordinate is the location of the Critter when it reaches the destination. |
| Message format: | REACHED_DESTINATION:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | REACHED_DESTINATION:{X=145,Y=352} |

| | |
|---|---|
| LAUNCH | This message is sent when a Critter is started in a level. The received coordinate is the starting location of the Critter.<br><br>**Before receiving this message, the Critter controller must not send any requests to the Critterworld environment.** |
| Message format: | LAUNCH:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | LAUNCH:{X=145,Y=352} |

| | |
|---|---|
| SHUTDOWN | This message is sent when a Critter is shut down by the Critterworld environment, perhaps because a level ends or the game is over. The received coordinate is the location of the Critter at the point where SHUTDOWN was received.<br><br>**After receiving this message, the Critter controller must not send any further requests to the Critterworld environment until a LAUNCH message is received, and any running threads should be terminated.** |
| Message format: | SHUTDOWN:<coordinate><br><br>Where:<br><br><coordinate> is {X=<x-coord>,Y=<y-coord>}<br><x-coord> is a horizontal pixel position in the arena<br><y-coord> is a vertical pixel position in the arena |
| Example: | SHUTDOWN:{X=132,Y=12} |

## *Important Notes!*

1. **Include the following *using* statements at the top of each class you create:**

   This will ensure that the appropriate classes and structures are available to you.

   ```
   using CritterController;
   using System;
   using System.Drawing;
   using System.IO;
   ```

   *This is very important. If you do not do this step, your code will not be able to use any of the interfaces mentioned above.*

2. **You may need to explicitly build the project after making modifications to your Critter controllers. In some cases, they will not build automatically if you select Start.**