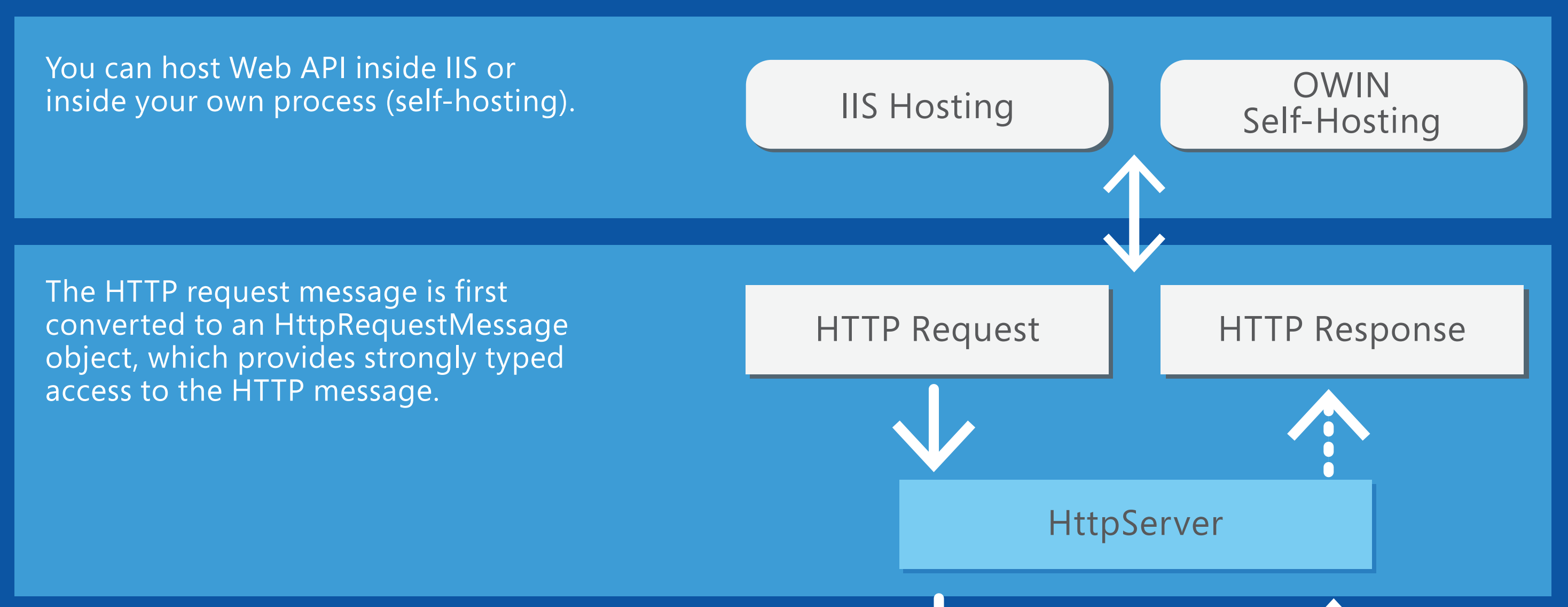
ASP.NET WEB API 2: HTTP MESSAGE LIFECYLE

Microsoft® .NET | ASP.NET

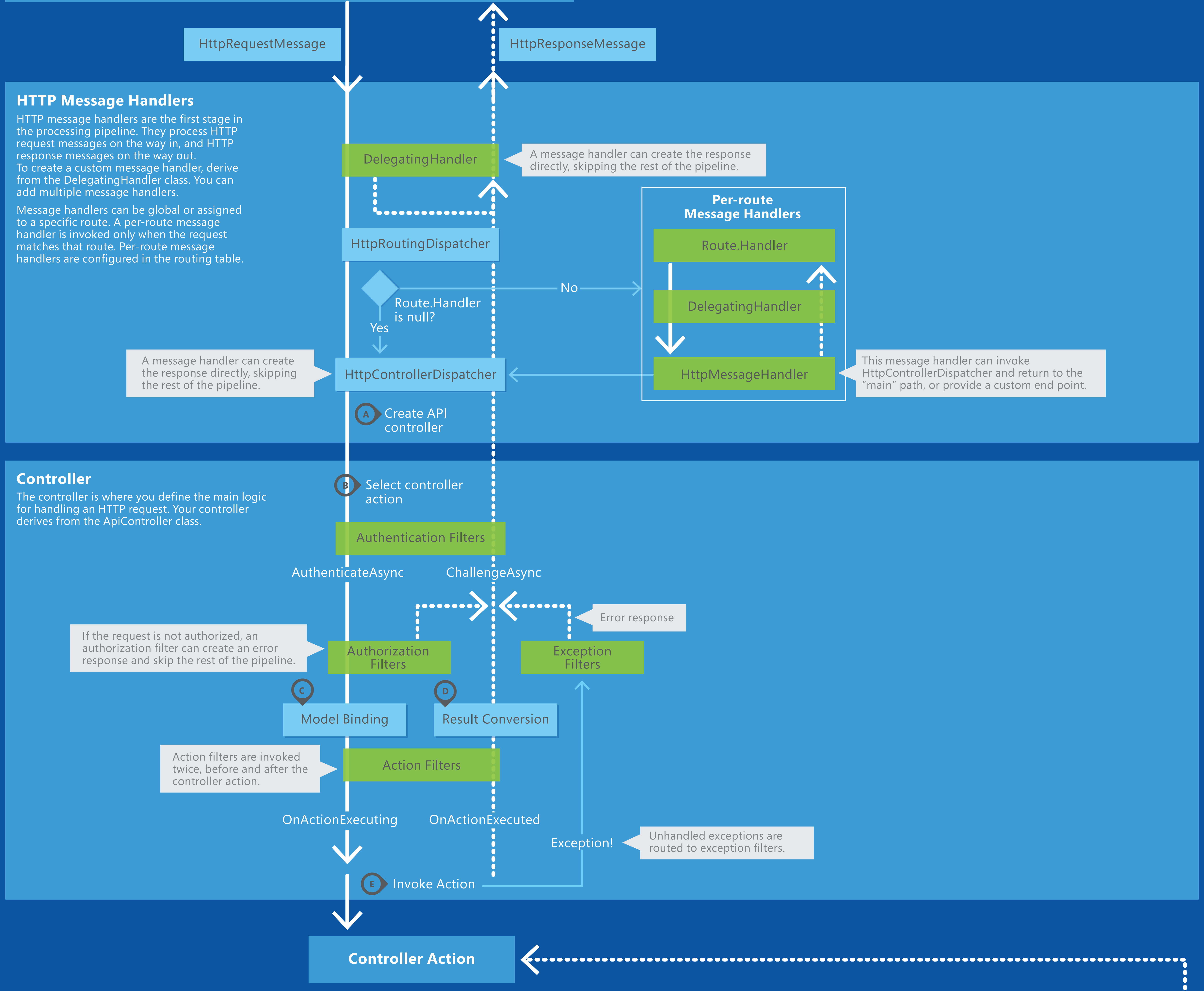
ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. It is an ideal platform for building RESTful applications on the .NET Framework.

This poster shows how an HTTP request flows through the Web API pipeline, and how the HTTP response flows back. The diagram also shows extensibility points, where you can add custom code or even replace the default behavior entirely. You can find documentation and tutorials for ASP.NET Web API at http://www.asp.net/web-api.



You can host Web API inside IIS or inside your own process (self-hosting).

The HTTP request message is first converted to an HttpRequestMessage object, which provides strongly typed access to the HTTP message.



HTTP Message Handlers

HTTP message handlers are the first stage in the processing pipeline. They process HTTP request messages on the way in, and HTTP response messages on the way out.

To create a custom message handler, derive from the DelegatingHandler class. You can add multiple message handlers.

Message handlers can be global or assigned to a specific route. A per-route message handler is invoked only when the request matches that route. Per-route message handlers are configured in the routing table.

A message handler can create the response directly, skipping the rest of the pipeline.

A message handler can create the response directly, skipping the rest of the pipeline.

This message handler can invoke HttpControllerDispatcher and return to the "main" path, or provide a custom end point.

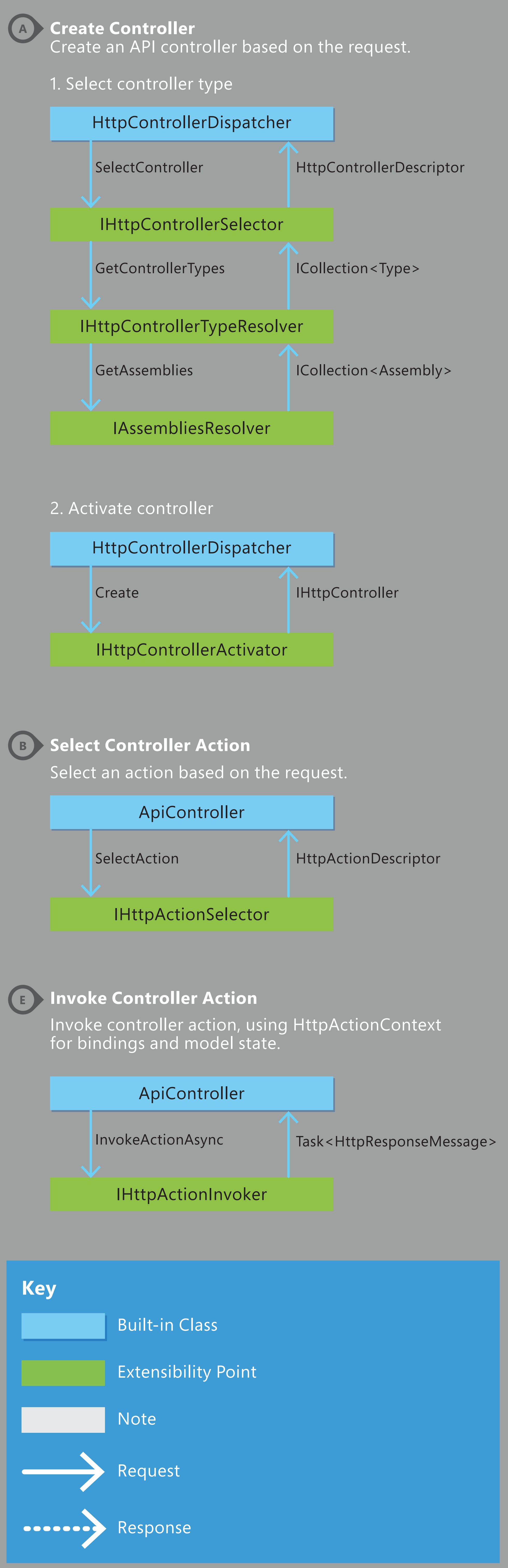
Controller

The controller is where you define the main logic for handling an HTTP request. Your controller derives from the ApiController class.

If the request is not authorized, an authorization filter can create an error response and skip the rest of the pipeline.

Action filters are invoked twice, before and after the controller action.

Unhandled exceptions are routed to exception filters.



A Create Controller
Create an API controller based on the request.

1. Select controller type

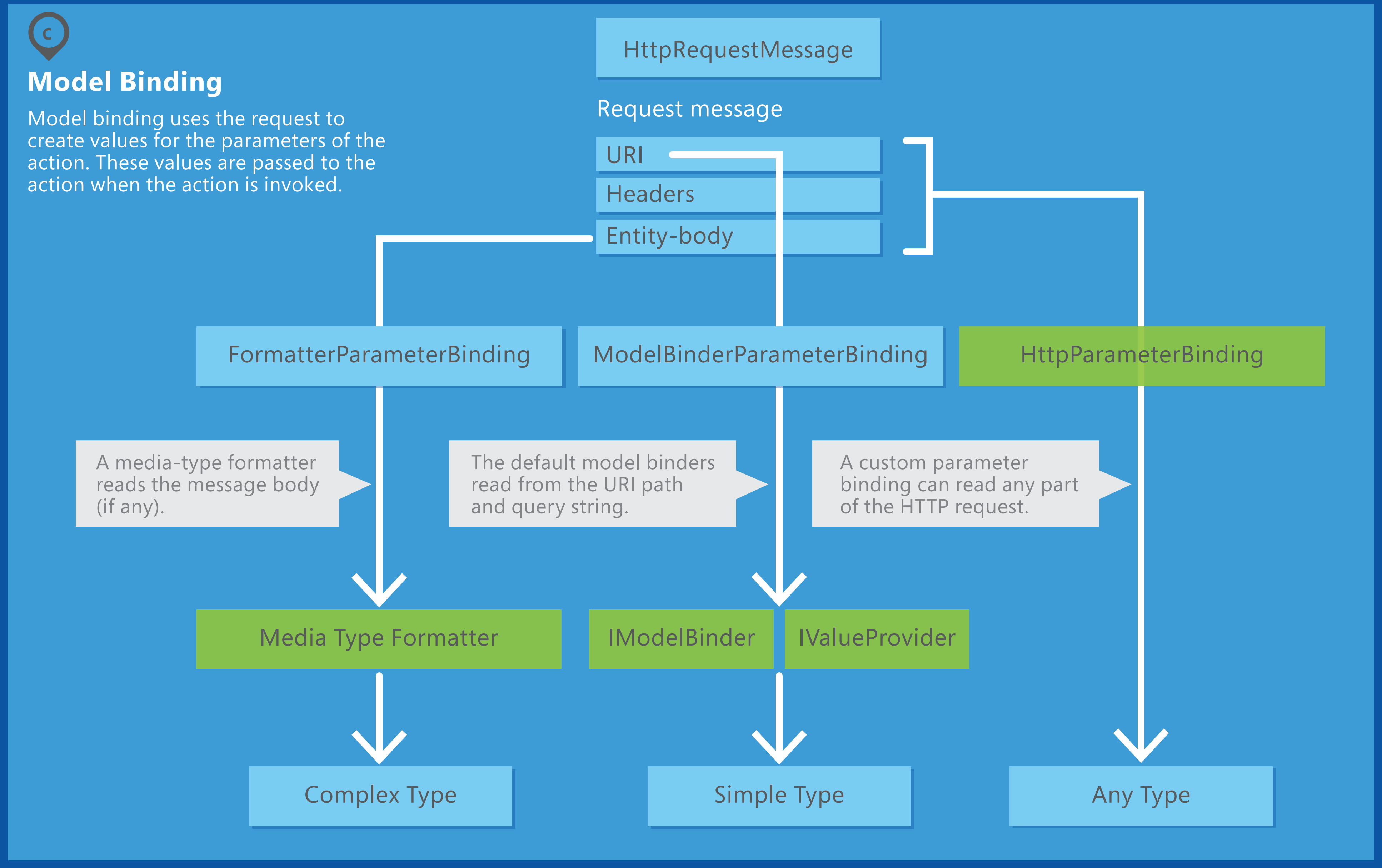
2. Activate controller

B Select Controller Action
Select an action based on the request.

E Invoke Controller Action
Invoke controller action, using HttpActionContext for bindings and model state.

Key

- ▭ Built-in Class
- 🟩 Extensibility Point
- ▭ Note
- ⟶ Request
- ⇢ Response



C

Model Binding

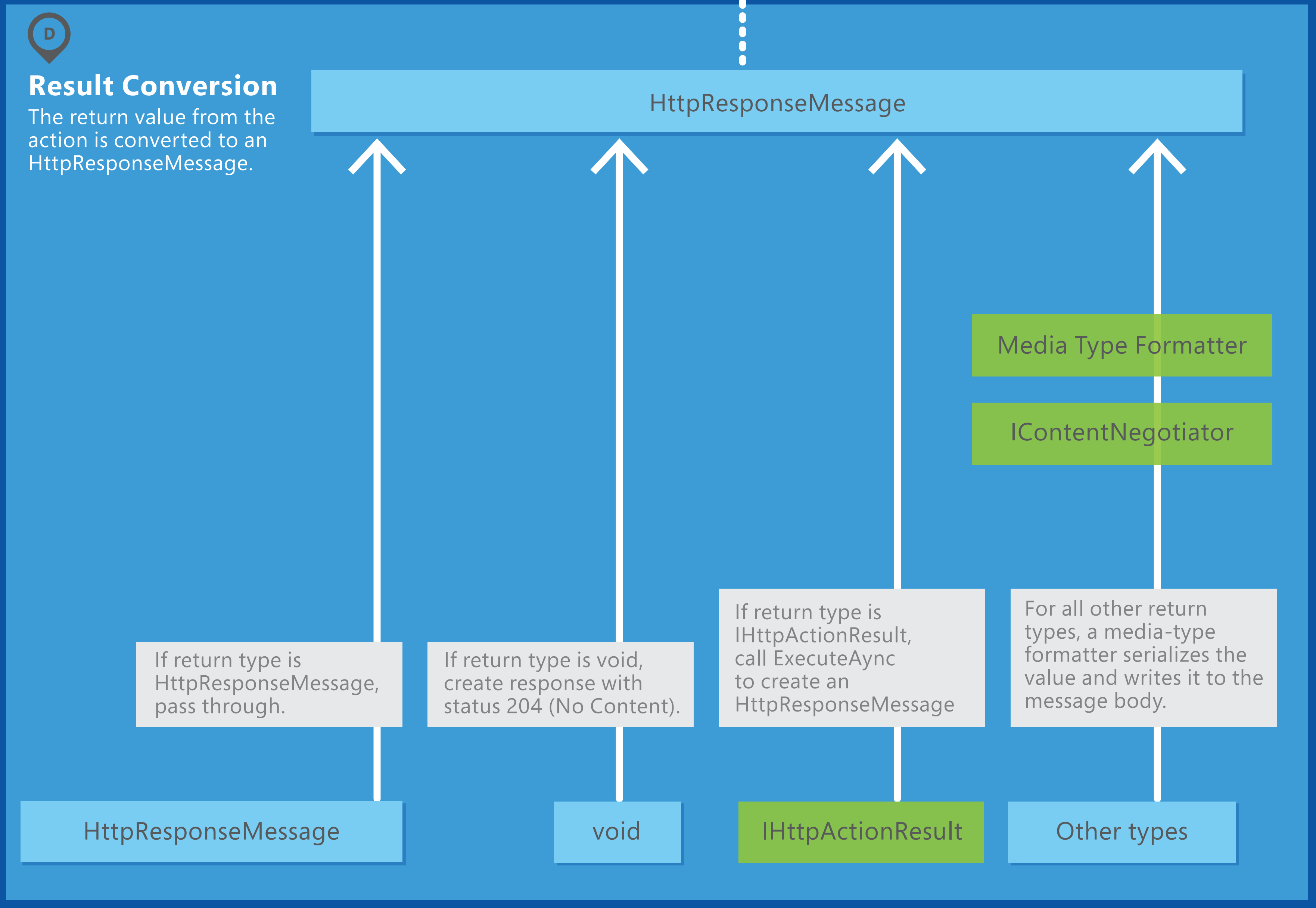
Model binding uses the request to create values for the parameters of the action. These values are passed to the action when the action is invoked.

A media-type formatter reads the message body (if any).

The default model binders read from the URI path and query string.

A custom parameter binding can read any part of the HTTP request.

Action parameters



D

Result Conversion

The return value from the action is converted to an HttpResponseMessage.

If return type is HttpResponseMessage, pass through.

If return type is void, create response with status 204 (No Content).

If return type is IHttpActionResult, call ExecuteAsync to create an HttpResponseMessage

For all other return types, a media-type formatter serializes the value and writes it to the message body.

Action return value

Microsoft

Email: MSPoster@microsoft.com