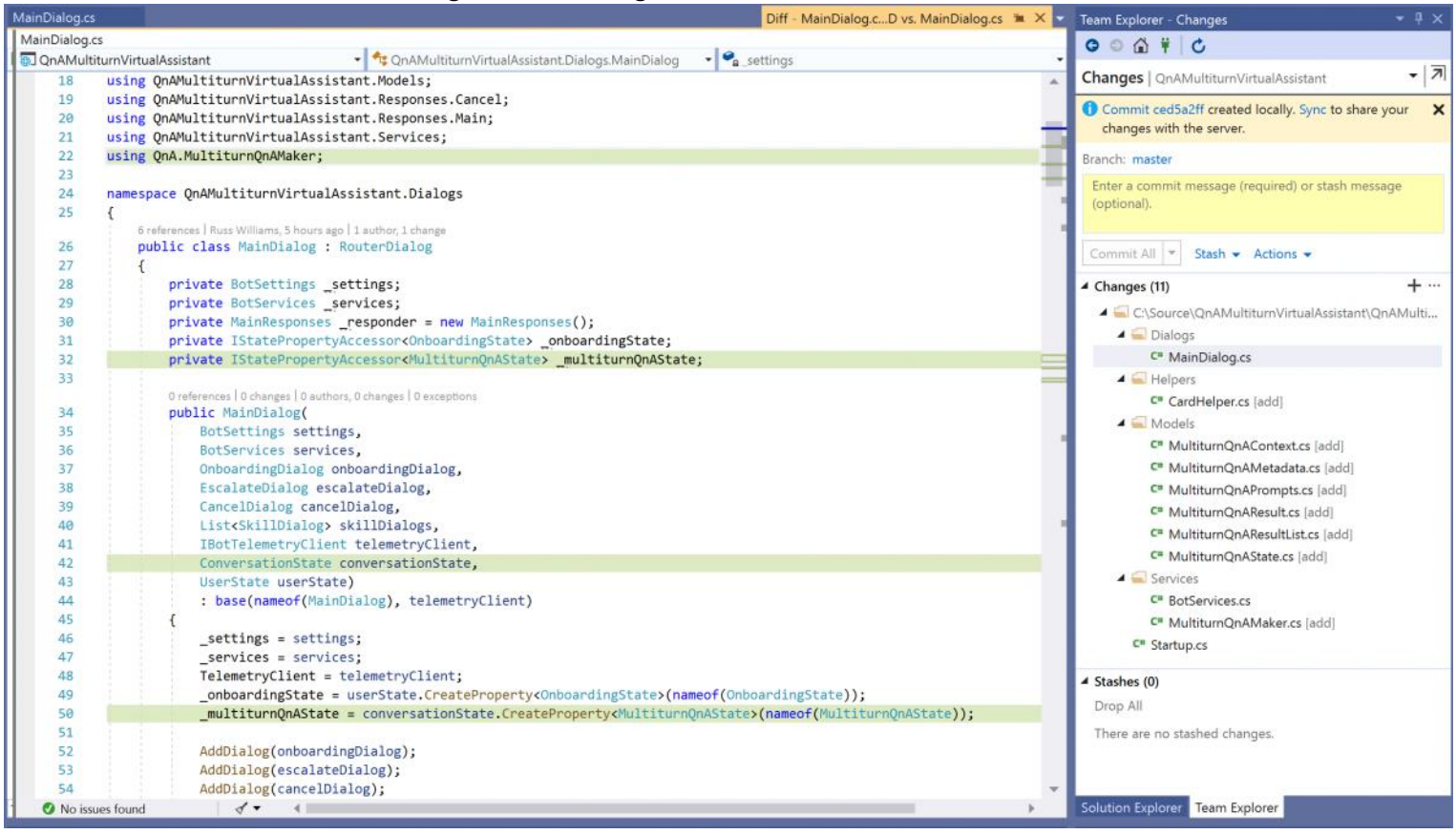
Changes to MainDialog.cs

Thursday, June 13, 2019 1:01 AM

There are two sets of changes you need to make to the MainDialog.cs file.

Diff screen shot to show first set of changes to MainDialog.cs



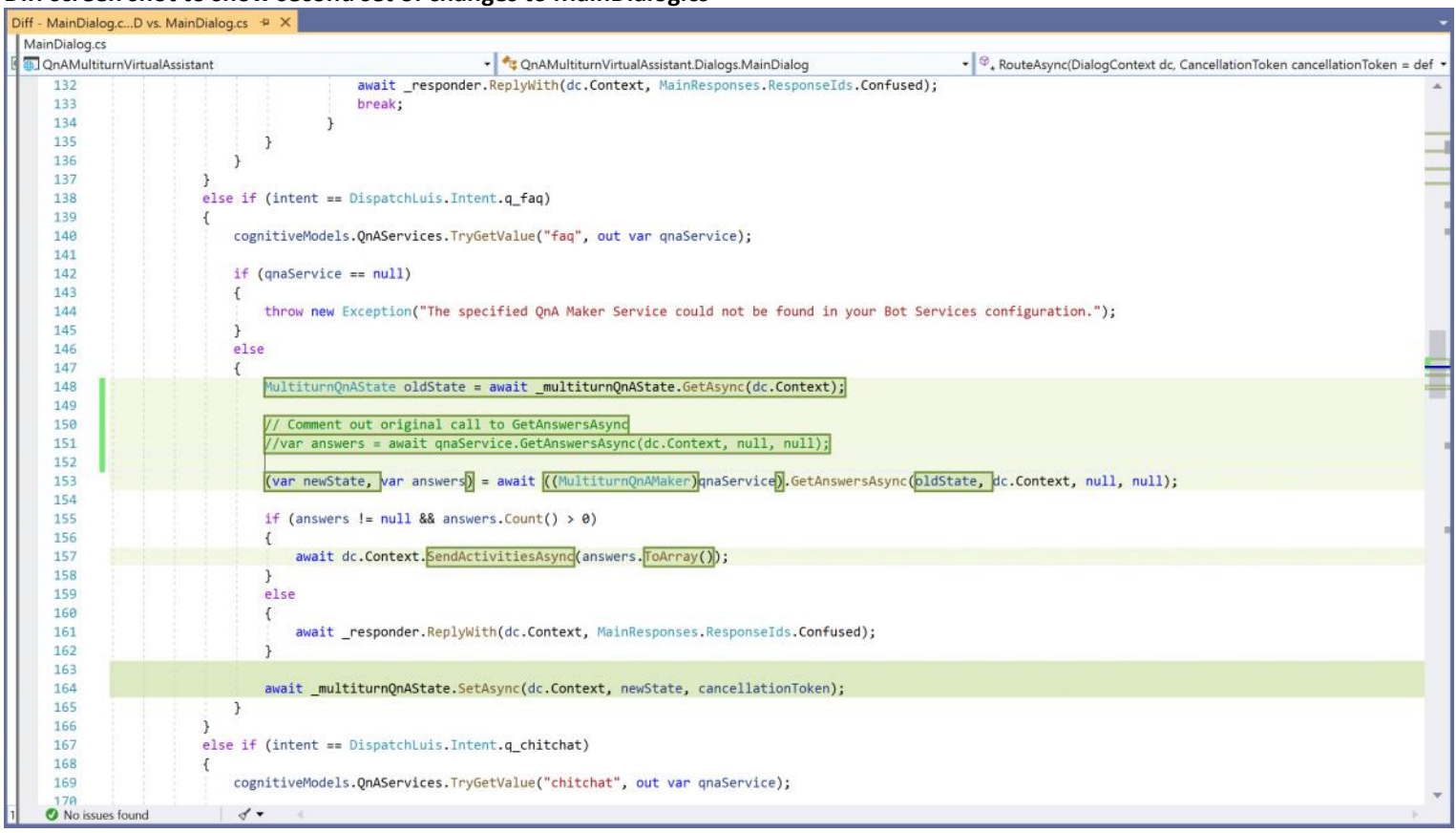
Snippet for easy cut & paste (copy bolded lines)

using QnA.MultiturnQnAMaker;

```
namespace QnAMultiturnVirtualAssistant.Dialogs
{
  public class MainDialog : RouterDialog
  {
    private BotSettings _settings;
    private BotServices _services;
    private MainResponses _responder = new MainResponses();
    private IStatePropertyAccessor<OnboardingState> _onboardingState;
    private IStatePropertyAccessor<MultiturnQnAState> _multiturnQnAState;

    public MainDialog(
      BotSettings settings,
      BotServices services,
      OnboardingDialog onboardingDialog,
      EscalateDialog escalateDialog,
      CancelDialog cancelDialog,
      List<SkillDialog> skillDialogs,
      IBotTelemetryClient telemetryClient,
      ConversationState conversationState,
      UserState userState)
      : base(nameof(MainDialog), telemetryClient)
    {
      _settings = settings;
      _services = services;
      TelemetryClient = telemetryClient;
      _onboardingState = userState.CreateProperty<OnboardingState>(nameof(OnboardingState));
      _multiturnQnAState = conversationState.CreateProperty<MultiturnQnAState>(nameof(MultiturnQnAState));
```

Diff screen shot to show second set of changes to MainDialog.cs




```
                                await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
                                break;
                            }
                        }
                    }
                }
                else if (intent == DispatchLuis.Intent.q_faq)
                {
                    cognitiveModels.QnAServices.TryGetValue("faq", out var qnaService);

                    if (qnaService == null)
                    {
                        throw new Exception("The specified QnA Maker Service could not be found in your Bot Services configuration.");
                    }
                    else
                    {
                        MultiturnQnAState oldState = await _multiturnQnAState.GetAsync(dc.Context);

                        // Comment out original call to GetAnswersAsync
                        //var answers = await qnaService.GetAnswersAsync(dc.Context, null, null);

                        (var newState, var answers) = await ((MultiturnQnAMaker)qnaService).GetAnswersAsync(oldState, dc.Context, null, null);

                        if (answers != null && answers.Count() > 0)
                        {
                            await dc.Context.SendActivitiesAsync(answers.ToArray());
                        }
                        else
                        {
                            await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
                        }

                        await _multiturnQnAState.SetAsync(dc.Context, newState, cancellationToken);
                    }
                }
                else if (intent == DispatchLuis.Intent.q_chitchat)
                {
                    cognitiveModels.QnAServices.TryGetValue("chitchat", out var qnaService);
```

Snippet for easy cut & paste (copy bolded lines)

```
else if (intent == DispatchLuis.Intent.q_faq)
{
   cognitiveModels.QnAServices.TryGetValue("faq", out var qnaService);

   if (qnaService == null)
   {
      throw new Exception("The specified QnA Maker Service could not be found in your Bot Services configuration.");
   }
   else
   {
      MultiturnQnAState oldState = await _multiturnQnAState.GetAsync(dc.Context);

      // Comment out original call to GetAnswersAsync
      //var answers = await qnaService.GetAnswersAsync(dc.Context, null, null);

      (var newState, var answers) = await ((MultiturnQnAMaker)qnaService).GetAnswersAsync(oldState, dc.Context, null, null);

      if (answers != null && answers.Count() > 0)
      {
         await dc.Context.SendActivitiesAsync(answers.ToArray());
      }
      else
      {
         await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
      }

      await _multiturnQnAState.SetAsync(dc.Context, newState, cancellationToken);
   }
}
```