ENv2（ExposureWindow mode）

対応コード取り込みについて

2021/11/09

なにをやっているか

接触確認API バージョンアップ #300

Open **kazgoto** opened this issue on Aug 3 · 1 comment · May be fixed by #381

kazgoto commented on Aug 3 • edited by cocoa-dev　Collaborator

その機能リクエストは何らかの問題に関連しますか / Is your feature request related to a problem?

本Issueの作成時点でCOCOAが利用している接触確認APIは、「Legacy V1」と呼ばれるバージョンを利用していますが、これは非推奨となっており、新しいバージョン（ExposureWindow mode, ENv2）への移行が強く推奨されています。

https://developers.google.com/android/exposure-notifications/exposure-notifications-api

> We strongly recommend migrating to ExposureWindow mode. This mode allows you to separately view and revise or revoke matches from multiple days, while still leaving enough quota to update six times per day. Although the legacy v1 mode has more quota that can be used to partition matches by day, your app wastes the user's battery to do so and can even sometimes run out of quota.

解決策についてお書きください / Describe the solution you'd like

COCOAをExposureWindow mode（iOSではENv2）に移行します。

https://github.com/cocoa-mhlw/cocoa/issues/300

feature/cocoa2

https://github.com/cocoa-mhlw/cocoa/pull/381

Xamarin.ExposureNotification → Cappuccino

Xamarin.ExposureNotificationを削除

Cappuccinoに必要なNuGetパッケージをTempNugetFeedに格納

- Chino.Android.[version].nupkg
- Chino.Common.[version].nupkg
- Chino.iOS.[verison].nupkg
- XamarinComponents（変更点）
  - Xamarin.GooglePlayServices.Nearby.ExposureNotification.[version].nupkg
  - Xamarin.iOS.ExposureNotification.[version].nupkg

https://github.com/keiji/chino

Xamarin.ExposureNotification → Cappuccino

削除したクラス（Xamarin.ExposureNotificaiton関係）

- Covid19Radar/Covid19Radar/Services/ExposureNotificationHandler.cs
- Covid19Radar/Covid19Radar/Services/ExposureNotificationService.cs
- Covid19Radar/Covid19Radar/Services/ExposureNotificationStatusService.cs
- Covid19Radar/Covid19Radar/Services/ExposureNotificationStatusServiceMock.cs
- Covid19Radar/Covid19Radar/Services/IExposureNotificationStatusService.cs
- Covid19Radar/Covid19Radar.iOS/Services/ExposureNotificationStatusPlatformService.cs
- Covid19Radar/Tests/Covid19Radar.UnitTests/Services/ExposureNotificationServiceTests.cs
- Covid19Radar/Tests/Covid19Radar.UnitTests/Services/ExposureNotificationStatusServiceTests.cs

接触確認機能

接触確認機能の有効・無効化

```
public Command OnClickEnable => new Command(async () =>
{
    loggerService.StartMethod();

    try
    {
        await exposureNotificationApiService.StartExposureNotificationAsync();
        await NavigationService.NavigateAsync(nameof(TutorialPage6));
    }
    catch (ENException exception)
    {
        loggerService.Exception("ENException", exception);
        await NavigationService.NavigateAsync(nameof(TutorialPage6));
    }
    finally
    {
        loggerService.EndMethod();
    }
});
```

Covid19Radar.ViewModels.TutorialPage4ViewModel

Androidの場合

```
public readonly ExposureNotificationClient Client = new ExposureNotificationClient();

public override async Task StartAsync()
{
    try
    {
        await Client.StartAsync();
    }
    catch (ApiException apiException)
    {
        if (apiException.StatusCode == CommonStatusCodes.ResolutionRequired)
        {
            apiException.Status.StartResolutionForResult(Platform.CurrentActivity, REQUEST_EN_START);
            throw new ENException(ENException.Code_Android.FAILED_UNAUTHORIZED,
                "StartAsync StartResolutionForResult");
        }
        else
        {
            throw apiException;
        }
    }
}
```

Covid19Radar.Droid.Services.ExposureNotificationApiService

OnActivityResultの処理

```
protected override void OnActivityResult(int requestCode,
                                         Result resultCode,
                                         Intent data
)
{
    base.OnActivityResult(requestCode, resultCode, data);

    var isOk = (resultCode == Result.Ok);

    FireExposureNotificationEvent(requestCode, isOk);
}
```

Covid19Radar.Droid.MainActivity

現在のページに結果を伝える

```
private void FireExposureNotificationEvent(int requestCode, bool isOk)
{
    Action<IExposureNotificationEventCallback> action = requestCode switch
    {
        ExposureNotificationApiService.REQUEST_EN_START
            => new Action<IExposureNotificationEventCallback>(callback =>
            {
                if(isOk)
                {
                    callback.OnEnabled();
                }
                else
                {
                    callback.OnDeclined();
                }
            }),
        ExposureNotificationApiService.REQUEST_GET_TEK_HISTORY
            => new Action<IExposureNotificationEventCallback>(callback => { callback.OnGetTekHistoryAllowed(); }),
        ExposureNotificationApiService.REQUEST_PREAUTHORIZE_KEYS
            => new Action<IExposureNotificationEventCallback>(callback => { callback.OnPreauthorizeAllowed(); }),
        _ => new Action<IExposureNotificationEventCallback>(callback => { /* do nothing */ }),
    };
    PageUtilities.InvokeViewAndViewModelAction(PageUtilities.GetCurrentPage(AppInstance.MainPage), action);
}
```

Covid19Radar.Droid.MainActivity

コールバックを受けて再度実行

```
public class TutorialPage4ViewModel : ViewModelBase, IExposureNotificationEventCallback
{

    …

    public async void OnEnabled()
    {
        loggerService.StartMethod();

        await exposureNotificationApiService.StartExposureNotificationAsync();
        await NavigationService.NavigateAsync(nameof(TutorialPage6));

        loggerService.EndMethod();
    }
}
```

Covid19Radar.ViewModels.TutorialPage4ViewModel

TEKの取得

```
private async Task SubmitDiagnosisKeys()
{
    loggerService.Info($"Submit DiagnosisKeys.");

    try
    {
        List<TemporaryExposureKey> temporaryExposureKeyList
            = await exposureNotificationApiService.GetTemporaryExposureKeyHistoryAsync();

        loggerService.Info($"TemporaryExposureKeys-count: {temporaryExposureKeyList.Count()}");

        IList<TemporaryExposureKey> filteredTemporaryExposureKeyList
            = TemporaryExposureKeyUtils.FiilterTemporaryExposureKeys(
                temporaryExposureKeyList,
                _diagnosisDate,
                AppConstants.DaysToSendTek,
                loggerService
                );

        loggerService.Info($"FilteredTemporaryExposureKeys-count: {filteredTemporaryExposureKeyList.Count()}");
```

Covid19Radar.ViewModels.NotifyOtherPageViewModel

TEKの取得（Androidの場合）

```
public async void OnGetTekHistoryAllowed()
{
    loggerService.StartMethod();

    await SubmitDiagnosisKeys();

    loggerService.EndMethod();
}
```

Covid19Radar.ViewModels.NotifyOtherPageViewModel

接触確認

```
public async Task ExposureDetectionAsync(CancellationTokenSource cancellationTokenSource = null)
{
            …

            await _exposureNotificationApiService.ProvideDiagnosisKeysAsync(
                 downloadedFileNameList,
                 exposureConfiguration,
                 cancellationTokenSource
                 );

             // Save LastProcessDiagnosisKeyTimestamp after ProvideDiagnosisKeysAsync was succeeded.
             var latestProcessTimestamp = targetDiagnosisKeyEntryList
                 .Select(diagnosisKeyEntry => diagnosisKeyEntry.Created)
                 .Max();
             await _userDataRepository.SetLastProcessDiagnosisKeyTimestampAsync(region, latestProcessTimestamp);

        }
        finally
        {
            RemoveFiles(downloadedFileNameList);
        }
    }
}
```

Covid19Radar.Services.AbsExposureDetectionBackgroundService

接触確認結果のコールバック（Android）

```
public class MainApplication : Application, IExposureNotificationHandler
{
    private Lazy<AbsExposureNotificationApiService> _exposureNotificationApiService
        = new Lazy<AbsExposureNotificationApiService>(() =>
        ServiceLocator.Current.GetInstance<AbsExposureNotificationApiService>());

    private Lazy<IExposureDetectionService> _exposureDetectionService
        = new Lazy<IExposureDetectionService>(() =>
        ServiceLocator.Current.GetInstance<IExposureDetectionService>());

    public AbsExposureNotificationClient GetEnClient()
    {
        if (_exposureNotificationApiService.Value is ExposureNotificationApiService exposureNotificationApiService)
        {
            return exposureNotificationApiService.Client;
        }
        else
        {
            return null;
        }
    }
```

Covid19Radar.Droid.MainApplication

```
public override void OnCreate()
{
    base.OnCreate();

    App.InitializeServiceLocator(RegisterPlatformTypes);

    AbsExposureNotificationClient.Handler = this;
    if (_exposureNotificationApiService.Value is ExposureNotificationApiService exposureNotificationApiService)
    {
        SetupENClient(exposureNotificationApiService.Client);
    }
}

private void SetupENClient(ExposureNotificationClient client)
{
    client.Init(this);
    client.ExposureDetectedV1JobSetting = _exposureDetectedV1JobSetting;
    client.ExposureDetectedV2JobSetting = _exposureDetectedV2JobSetting;
    client.ExposureNotDetectedJobSetting = _exposureNotDetectedJobSetting;
}
```

Covid19Radar.Droid.MainApplication

```
public void PreExposureDetected()
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.PreExposureDetected(exposureConfiguration, enVersion);
}
public void ExposureDetected(IList<DailySummary> dailySummaries, IList<ExposureWindow> exposureWindows)
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureDetected(exposureConfiguration, enVersion, dailySummaries, exposureWindows);
}
public void ExposureDetected(ExposureSummary exposureSummary, IList<ExposureInformation> exposureInformations)
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureDetected(exposureConfiguration, enVersion, exposureSummary, exposureInformations);
}
public void ExposureNotDetected()
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureNotDetected(exposureConfiguration, enVersion);
}
```

Covid19Radar.Droid.MainApplication

接触確認結果のコールバック（iOS）

```
    public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate, IExposureNotificationHandler
    {
        private Lazy<AbsExposureNotificationApiService> _exposureNotificationClient
            = new Lazy<AbsExposureNotificationApiService>(() =>
ServiceLocator.Current.GetInstance<AbsExposureNotificationApiService>());

       private Lazy<IExposureDetectionService> _exposureDetectionService
            = new Lazy<IExposureDetectionService>(() => ServiceLocator.Current.GetInstance<IExposureDetectionService>());

        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            …

            InitializeExposureNotificationClient();

            …
        }
```

Covid19Radar.iOS.AppDelegate

```
        public AbsExposureNotificationClient GetEnClient() => _exposureNotificationClient.Value;

        private void InitializeExposureNotificationClient()
        {
            AbsExposureNotificationClient.Handler = this;

            if (GetEnClient() is ExposureNotificationApiService exposureNotificationApiService)
            {
                exposureNotificationApiService.UserExplanation = AppResources.LocalNotificationDescription;

#if DEBUG
                exposureNotificationApiService.IsTest = true;
#else
                exposureNotificationApiService.IsTest = false;
#endif
            }
        }
```

Covid19Radar.iOS.AppDelegate

```
public void PreExposureDetected()
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.PreExposureDetected(exposureConfiguration, enVersion);
}
public void ExposureDetected(IList<DailySummary> dailySummaries, IList<ExposureWindow> exposureWindows)
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureDetected(exposureConfiguration, enVersion, dailySummaries, exposureWindows);
}
public void ExposureDetected(ExposureSummary exposureSummary, IList<ExposureInformation> exposureInformations)
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureDetected(exposureConfiguration, enVersion, exposureSummary, exposureInformations);
}
public void ExposureNotDetected()
{
    var exposureConfiguration = GetEnClient().ExposureConfiguration;
    var enVersion = GetEnClient().GetVersionAsync()
        .GetAwaiter().GetResult().ToString();
    _exposureDetectionService.Value.ExposureNotDetected(exposureConfiguration, enVersion);
}
```

Covid19Radar.iOS.AppDelegate

エラーハンドリング

```
public class ENException : Exception
{
    // https://developers.google.com/android/reference/com/google/android/gms/nearby/exposurenotification/ExposureNotificationStatusCodes
    public static class Code_Android
    {

        /// <summary>
        /// The operation failed, without any more information.
        /// </summary>
        public const int FAILED = (int)PlatformType.Android + 0;

        …
    }

    // https://developer.apple.com/documentation/exposurenotification/enerror
    public static class Code_iOS
    {

        /// <summary>
        /// The API use is incorrect.
        /// </summary>
        public const int ApiMisuse = (int)PlatformType.iOS + 0;

        …
    }
}
```

https://github.com/keiji/chino/blob/master/Chino.Common/ExposureNotificationStatus.cs

状態ハンドリング

```
public class ExposureNotificationStatus
{

    // https://developers.google.com/android/reference/com/google/android/gms/nearby/exposurenotification/ExposureNotificationStatus
    public static class Code_Android
    {
        /// <summary>
        /// Exposure notification is running.
        /// </summary>
        public const int ACTIVATED = (int)PlatformType.Android + 0;

        …
    }

    // https://developer.apple.com/documentation/exposurenotification/enstatus
    public static class Code_iOS
    {
        /// <summary>
        /// Notification is active.
        /// </summary>
        public const int Active = (int)PlatformType.iOS + 0;

        …
    }
}
```

https://github.com/keiji/chino/blob/master/Chino.Common/ENException.cs

状態ハンドリング

```
public Command OnClickCheckStopReason => new Command(async () =>
{
   var statusCodes = await exposureNotificationApiService.GetStatusCodesAsync();

    if (statusCodes.Contains(ExposureNotificationStatus.Code_Android.BLUETOOTH_DISABLED)
    || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.BluetoothOff)
    )
    {
        bool isOK = await dialogService.ShowBluetoothOffWarningAsync();
        if (isOK)
        {
            externalNavigationService.NavigateBluetoothSettings();
        }
    }
    else if (statusCodes.Contains(ExposureNotificationStatus.Code_Android.LOCATION_DISABLED))
    {
        bool isOK = await dialogService.ShowLocationOffWarningAsync();
        if (isOK)
        {
            externalNavigationService.NavigateLocationSettings();
        }
    }
```

Covid19Radar.ViewModels.HomePageViewModel

状態ハンドリング

```
    else if (statusCodes.Contains(ExposureNotificationStatus.Code_Android.INACTIVATED)
    || statusCodes.Contains(ExposureNotificationStatus.Code_Android.FOCUS_LOST)
    || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.Disabled)
    || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.Unauthorized)
    )
    {
        bool isOK = await dialogService.ShowExposureNotificationOffWarningAsync();
        if (isOK)
        {
            externalNavigationService.NavigateAppSettings();
        }
    }

    loggerService.EndMethod();
});
```

Covid19Radar.ViewModels.HomePageViewModel

状態ハンドリング

```
private async Task UpdateView()
{
    loggerService.StartMethod();

    var daysOfUse = _userDataRepository.GetDaysOfUse();

    PastDate = daysOfUse.ToString();

    var statusCodes = await exposureNotificationApiService.GetStatusCodesAsync();

    var isStopped =
        statusCodes.Contains(ExposureNotificationStatus.Code_Android.INACTIVATED)
        || statusCodes.Contains(ExposureNotificationStatus.Code_Android.FOCUS_LOST)
        || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.Disabled)
        || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.Unauthorized)
        || statusCodes.Contains(ExposureNotificationStatus.Code_Android.BLUETOOTH_DISABLED)
        || statusCodes.Contains(ExposureNotificationStatus.Code_iOS.BluetoothOff)
        || statusCodes.Contains(ExposureNotificationStatus.Code_Android.LOCATION_DISABLED);

    …
```

Covid19Radar.ViewModels.HomePageViewModel

バックグラウンド処理

- Cappuccinoは担当しない
- `IBackgroundService`を実装する
- 接触確認のバックグラウンド処理は`AbsExposureDetectionBackgroundService`が担当
- 各プラットフォーム（Android/iOS）向けにサービスとして登録する

バックグラウンドサービス（Android）

```
 public override void Schedule()
 {
    WorkManager workManager = WorkManager.GetInstance(Platform.AppContext);

     PeriodicWorkRequest periodicWorkRequest = CreatePeriodicWorkRequest();
     workManager.EnqueueUniquePeriodicWork(
         CURRENT_WORK_NAME,
         ExistingPeriodicWorkPolicy.Keep,
         periodicWorkRequest
         );
}

 private static PeriodicWorkRequest CreatePeriodicWorkRequest()
 {
     var workRequestBuilder = new PeriodicWorkRequest.Builder(
         typeof(BackgroundWorker),
         INTERVAL_IN_MINUTES, TimeUnit.Minutes)
         .SetConstraints(new Constraints.Builder()
            .SetRequiresBatteryNotLow(true)
            .SetRequiredNetworkType(NetworkType.Connected)
            .Build())
         .SetBackoffCriteria(BackoffPolicy.Linear, BACKOFF_DELAY_IN_MINUTES, TimeUnit.Minutes);
     return workRequestBuilder.Build();
  }
```

Covid19Radar.Droid.Services.ExposureDetectionBackgroundService

バックグラウンドサービス（iOS）

```
public override void Schedule()
{
    _loggerService.StartMethod();
    _loggerService.Debug($"BGTASK_IDENTIFIER: {BGTASK_IDENTIFIER}");

    var result = BGTaskScheduler.Shared.Register(BGTASK_IDENTIFIER, null, task =>
    {
        _loggerService.Info("Background task has been started.");

        ScheduleBgTask();

        var cancellationTokenSource = new CancellationTokenSource();
        task.ExpirationHandler = cancellationTokenSource.Cancel;

        _ = Task.Run(async () =>
        {
            try
            {
                IList<ExposureNotificationStatus> statuses = _exposureNotificationApiService.GetStatusesAsync()
                    .GetAwaiter().GetResult();
```

Covid19Radar.iOS.Services.ExposureDetectionBackgroundService

```
                bool isUnauthorized = statuses
                    .Where(status => status.Code == ExposureNotificationStatus.Code_iOS.Unauthorized)
                    .Count() != 0;
                if (isUnauthorized)
                {
                    _loggerService.Error("Exposure notofication is not authorized.");
                    task.SetTaskCompleted(true);
                    return;
                }

                await ExposureDetectionAsync(cancellationTokenSource);
                task.SetTaskCompleted(true);
            }
            catch (OperationCanceledException exception)
            {
                _loggerService.Exception($"Background task canceled.", exception);
                task.SetTaskCompleted(false);
            }
            catch (Exception exception)
            {
                _loggerService.Exception($"Exception", exception);
                task.SetTaskCompleted(false);
            }
            finally
            {
                cancellationTokenSource.Dispose();
            }
        }, cancellationTokenSource.Token);
    });
```

Covid19Radar.iOS.Services.ExposureDetectionBackgroundService

```
        if (result)
        {
            _loggerService.Debug("BGTaskScheduler.Shared.Register succeeded.");
        }
        else
        {
            _loggerService.Info("BGTaskScheduler.Shared.Register failed.");
        }

        ScheduleBgTask();

        _loggerService.EndMethod();
    }
```

Covid19Radar.iOS.Services.ExposureDetectionBackgroundService

ユーザーデータ（Repository）

Repositoryに集約

サーバーから情報を取得する処理もRepositoryが担当する

- DiagnosisKeyRepository → 診断キー
- ExposureConfigurationRepository → 接触確認設定
- ServerConfigurationRepository → 接続先サーバーの設定
- UserDataRepository → ユーザーデータ（接触情報含む）

接触確認設定（ExposureConfiguration）

```
"google_exposure_config": {
  "attenuation_scores": [ 4, 4, 4, 4, 4, 4, 4, 4 ],
  "attenuation_weight": 50,
  "days_since_last_exposure_scores": [ 4, 4, 4, 4, 4, 4, 4, 4 ],
  "days_since_last_exposure_weight": 50,
  "duration_at_attenuation_thresholds": [ 50, 74 ],
  "duration_scores":  [ 4, 4, 4, 4, 4, 4, 4, 4 ],
  "duration_weight": 50,
  "minimum_risk_score": 4,
  "transmission_risk_scores":  [ 4, 4, 4, 4, 4, 4, 4, 4 ],
  "transmission_risk_weight": 50
},
"apple_exposure_config_v1": {
  "attenuation_level_values": [ 1, 2, 3, 4, 5, 6, 7, 8 ],
  "days_since_last_exposure_level_values":  [ 1, 2, 3, 4, 5, 6, 7, 8 ],
  "duration_level_values":  [ 1, 2, 3, 4, 5, 6, 7, 8 ],
  "transmission_risk_level_values":  [ 1, 2, 3, 4, 5, 6, 7, 8 ],
  "minimum_risk_score": 0,
  "minimum_risk_score_full_range": 0.0
},
```

https://github.com/keiji/chino/wiki/Default-ExposureConfiguration

接触確認設定（ExposureConfiguration）


```
"google_diagnosis_keys_data_mapping_config": {
  "infectiousness_when_days_since_onset_missing": 1,
  "report_type_when_missing": 1,
  "infectiousness_for_days_since_onset_of_symptoms": {
    "-14": 0,
    "-13": 0,
    "-12": 0,
    "-11": 0,
    "-10": 0,
    "-9": 0,
    "-8": 0,
    "-7": 0,
    "-6": 0,
    "-5": 1,
    "-4": 1,
    "-3": 1,
    "-2": 2,
    "-1": 2,
    "0": 2,
    "1": 2,
    "2": 2,
    "3": 2,
    "4": 2,
    "5": 2,
    "6": 1,
    "7": 1,
    "8": 1,
    "9": 1,
    "10": 1,
    "11": 0,
    "12": 0,
    "13": 0,
    "14": 0
  }
},
```


https://github.com/keiji/chino/wiki/Default-ExposureConfiguration

接触確認設定（ExposureConfiguration）

```
"google_daily_summaries_config": {
  “attenuation_bucket_threshold_db": [ 50, 70, 90 ],
  "attenuation_bucket_weights": [ 1.0, 1.0, 1.0, 1.0 ],
  "days_since_exposure_threshold": 0,
  "infectiousness_weights": {
    "High": 1.0,
    "Standard": 1.0
  },
  "minimum_window_score": 0.0,
  "report_type_weights": {
    "ConfirmedClinicalDiagnosis": 1.0,
    "ConfirmedTest": 1.0,
    "SelfReport": 1.0,
    "Recursive": 1.0
  }
},
```

https://github.com/keiji/chino/wiki/Default-ExposureConfiguration

接触確認設定（ExposureConfiguration）


```
"apple_exposure_config_v2": {
  "infectiousness_when_days_since_onset_missing": 1,
  "attenuation_duration_thresholds": [ 50, 70, 90 ],
  “immediate_duration_weight": 100.0,
  "near_duration_weight": 100.0,
  "medium_duration_weight": 100.0,
  "other_duration_weight": 100.0,
  "DaysSinceLastExposureThreshold": 0,
  "infectiousness_for_days_since_onset_of_symptoms": {
    "-14": 1,
    "-13": 1,
    "-12": 1,
    "-11": 1,
    "-10": 1,
    "-9": 1,
    "-8": 1,
    "-7": 1,
    "-6": 1,
    "-5": 1,
    "-4": 1,
    "-3": 1,
    "-2": 1,
    "-1": 1,
    "0": 1,
    "1": 1,
    "2": 1,
    "3": 1,
    "4": 1,
    "5": 1,
    "6": 1,
    "7": 1,
    "8": 1,
    "9": 1,
    "10": 1,
    "11": 1,
    "12": 1,
    "13": 1,
    "14": 1
  },
  "infectiousness_high_weight": 100.0,
  "infectiousness_standard_weight": 100.0,
  "report_type_confirmed_clinical_diagnosis_weight": 100.0,
  "report_type_confirmed_test_weight": 100.0,
  "report_type_recursive_weight": 100.0,
  "report_type_self_reported_weight": 100.0,
  "report_type_none_map": 1
}
```

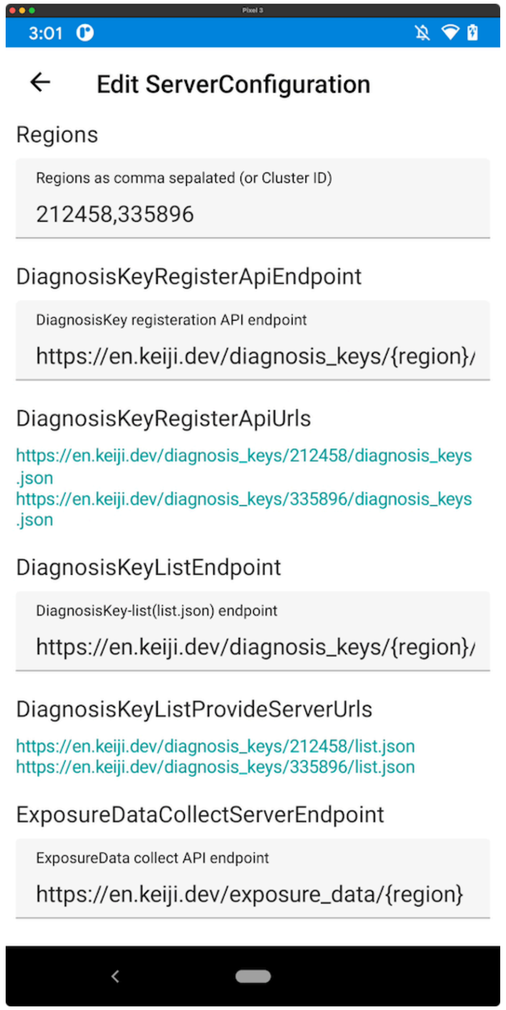

https://github.com/keiji/chino/wiki/Default-ExposureConfiguration

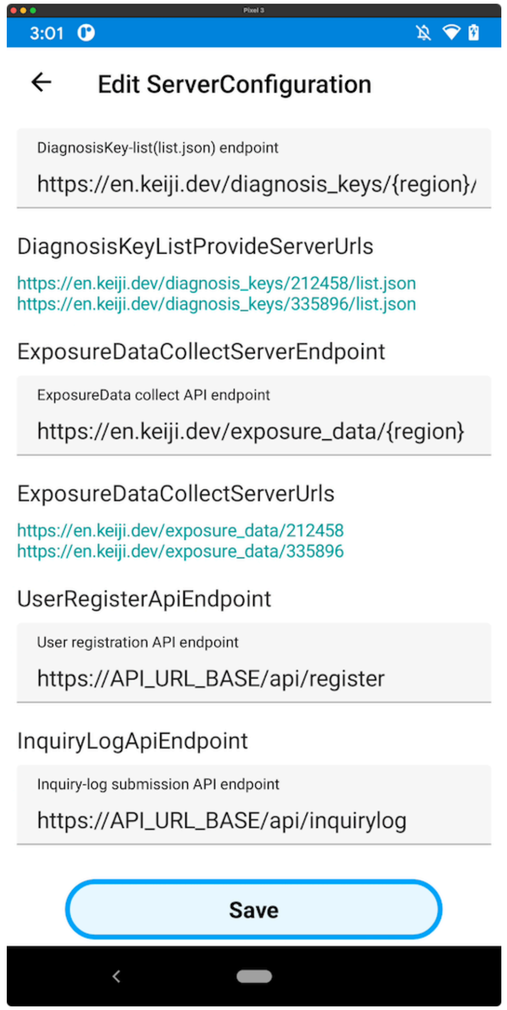
接触確認設定（ExposureConfiguration）

```
"google_daily_summaries_config": {
  "attenuation_bucket_threshold_db": [ 50, 70, 90 ],
  "attenuation_bucket_weights": [ 1.0, 1.0, 1.0, 1.0 ],
  "days_since_exposure_threshold": 0,
  "infectiousness_weights": {
    "High": 1.0,
    "Standard": 1.0
  },
  "minimum_window_score": 0.0,
  "report_type_weights": {
    "ConfirmedClinicalDiagnosis": 1.0,
    "ConfirmedTest": 1.0,
    "SelfReport": 1.0,
    "Recursive": 1.0
  }
},
```

https://github.com/keiji/chino/wiki/Default-ExposureConfiguration

デバッグ機能





https://github.com/cocoa-mhlw/cocoa/pull/438

陽性情報登録API（v3）

ENv2（ExposureWindow mode）に対応する新しいAPI

送信する情報

- 処理番号
- TEKs
- Region
- デバイス認証用の情報（Android/iOS）
- 症状が出た日・（無症状の場合）検査日
- IdempotencyKey（冪等キー）

https://github.com/cocoa-mhlw/cocoa/blob/feature/cocoa2/Covid19Radar/Covid19Radar/Services/DiagnosisKeyRegisterServer.cs

https://github.com/cocoa-mhlw/cocoa/blob/feature/cocoa2/src/Covid19Radar.Api/V3DiagnosisApi.cs

Protocol Buffers定義の変更

- 追加 revised_keys
- 削除 app_bundle_id
- 削除 android_package
- 追加 report_type
- 追加 days_since_onset_of_symptoms

https://github.com/cocoa-mhlw/cocoa/pull/381/files#diff-b04a283720693a2b20f648eb67bbf54ae5f4600c9e72dabf18d9b711ab2d0b94

Cosmos DB 変更点

ReportTypeとDaysSinceOnsetOfSymptomsのカラムを追加
（どちらもint型）

https://github.com/cocoa-mhlw/cocoa/pull/381/files#diff-b1287f5d9522dd31943016f47594650cc0bffc7fee0d46af610695ec72070337

今後の課題

ENv2（ExposureWindow mode）から得た接触情報（DailySummaries, ExposureWindows）をどのように表示するか検討中（野上さん主導）

- リスクレベルを多段階で表示
- ENv1（Legacy v1）時点で保存した接触情報も表示できる必要がある

（要検討）feature/cocoa2ブランチをENv1の状態でテスト・リリースする段階を踏んだ方がよいか？

その他COCOA2に入れたいこと

- すべての依存ライブラリを可能な限り最新バージョンにアップデート
- IPreferencesServiceの取り扱う型を厳密に扱う
  https://github.com/cocoa-mhlw/cocoa/issues/400
- サーバー側のAPIが廃止されるケースに対応する
  https://github.com/cocoa-mhlw/cocoa/issues/393
- アップデートの確認ファイルをAndroidとiOSで分離する
  https://github.com/cocoa-mhlw/cocoa/issues/392
- 接触情報の保存先をSecureStorageServiceから移管する
  https://github.com/cocoa-mhlw/cocoa/issues/374
- バックグラウンド処理からアプリのアップデートを通知でお知らせ
- ダミーリクエスト送信で陽性者だと特定しにくくする
- コントリビューター一覧ページを設定画面内に表示
- 動作ログの洗い替えをWorkManagerに切り換え