

Commit ID	Changes	Comments
1463a176bf52172560c3893c84b61c86e1b47f5	<pre>diff --git a/src/main/java/com/manh/cp/appointment/appointment/constants/FeatureFlagConstants.java b/src/main/java/com/manh/cp/appointment/appointment/constants/FeatureFlagConstants.java Index dd0b50b37..84bb7cd76 100644 --- a/src/main/java/com/manh/cp/appointment/appointment/constants/FeatureFlagConstants.java +++ b/src/main/java/com/manh/cp/appointment/appointment/constants/FeatureFlagConstants.java @@ -33,5 +33,5 @@ public class FeatureFlagConstants public static final String FF_24_4_FIX_EDIT_APPOINTMENT_DATE = "SUP-522225#2024-08"; - public static final String FF_99_9_FIX_FACILITY_DAY_LEVEL_SCOPE_FOR_NON_UTC_TIMEZONE = "SUP-504432#2024-08"; + public static final String FF_99_9_FIX_FACILITY_DAY_LEVEL_SCOPE_FOR_NON_UTC_TIMEZONE = "SUP-518897#2024-08"; }</pre>	
39c8f648fdb08e8717237ae4ffca5b181cd3fd33	<pre>diff --git a/gradle.properties b/gradle.properties index 63e41475e..541cd2205 100644 --- a/gradle.properties +++ b/gradle.properties @@ -1,1 +1 @@ -dbChecksum=5cf61ada06975e25d78b471e00b5184e +dbChecksum=153d228ee28c89f7182274a47fc7cd8e diff --git a/src/component-test/java/com/manh/cp/appointment/parameterizedTests/CapacityParameterizedTests.java b/src/component-test/java/com/manh/cp/appointment/parameterizedTests/CapacityParameterizedTests.java index 816a8d5b4..0a4648a8d 100644 --- a/src/component-test/java/com/manh/cp/appointment/parameterizedTests/CapacityParameterizedTests.java +++ b/src/component-test/java/com/manh/cp/appointment/parameterizedTests/CapacityParameterizedTests.java @@ -55,6 +55,14 @@ public class CapacityParameterizedTests extends ParameterizedTestBase { "test 11", "appointment/appointment/parameterizedTests/capacityTests/test11/" }, { "test 12", "appointment/appointment/parameterizedTests/capacityTests/test12/" }, { "test 13", "appointment/appointment/parameterizedTests/capacityTests/test13/" }, + { "test 14", "appointment/appointment/parameterizedTests/capacityTests/test14/" }, + { "test 15", "appointment/appointment/parameterizedTests/capacityTests/test15/" }, + { "test 16", "appointment/appointment/parameterizedTests/capacityTests/test16/" }, + { "test 17", "appointment/appointment/parameterizedTests/capacityTests/test17/" }, + { "test 18", "appointment/appointment/parameterizedTests/capacityTests/test18/" }, + { "test 19", "appointment/appointment/parameterizedTests/capacityTests/test19/" }, + { "test 20", "appointment/appointment/parameterizedTests/capacityTests/test20/" }, + { "test 21", "appointment/appointment/parameterizedTests/capacityTests/test21/" }, }); // @@ -88,8 +96,6 @@ public class CapacityParameterizedTests extends ParameterizedTestBase context = Context.renewContext(null, "system", "1", null, null, facilityId, null, null, "1");</pre>	<p>Question(may be to BA) : Can we have Day Level Capacity defined with both # of appointment as well as for appointment type?</p> <p>Same question as Day Level Capacity do we need to set scheduled Appt count as 1 ? I only increase the count to 1 if the appointment is scheduled. Otherwise, it will be created with the count as 0</p> <p>I'm really not sure why they wouldn't just create two separate records for that. What would your constraint type Id even look like in that scenario? That being said, it doesn't look like we have any validations for this field, as I was able to create a capacity record with the constraint type "garbage" @({5c3e1980d254b55db84d9cb4} - if they create 2 day level records one for #of appointments and one more for appt type , we are validating against both?</p> <p>@({6152e14e289a54006ab32626}) One day level utilization record will have all the information needed for both capacity checks, so I pull it once at the start and use it for all the capacity checks for that day review : we are checking for facility level capacity at 2 place. line@268 \(\here\ and line#290. Do we need at both place?</p> <p>do we need to check null for 'timeIntervalUtilization.getScheduledLiveUnloadAppts()' ? That's a fair point. If all the capacity checks pass, I can return the result of the facility capacity check. No reason to list it in two places with two conditions</p>

949be2ac0e957cfbe981da242384424b4786067a	<pre> diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentDaylightChangeParameterizedTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentDaylightChangeParameterizedTest.java index 92a3ad5d8..88421b0ca 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentDaylightChangeParameterizedTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentDaylightChangeParameterizedTest.java @@ -44,6 +44,7 @@ import com.manh.cp.appointment.appointment.TestUtil; import com.manh.cp.appointment.appointment.api.service.AppointmentMapService; import com.manh.cp.appointment.appointment.client.dto.calendar.CalendarGridIntervalCapacities; import com.manh.cp.appointment.appointment.client.dto.calendar.CalendarResponseDTO; +import com.manh.cp.appointment.appointment.constants.FeatureFlagConstants; import com.manh.cp.appointment.appointment.constants.FeatureIdentifier; import com.manh.cp.appointment.appointment.impl.service.AppointmentManager; import com.manh.cp.appointment.appointment.mocks.MockAppointmentUtil; @@ -55,6 +56,7 @@ import com.manh.cp.fw.core.logging.Logger; import com.manh.cp.fw.core.rest.RestApiResponse; import com.manh.cp.fw.dmttest.JsonUtil; import com.manh.cp.fw.entity.domain.entity.EntityList; +import com.manh.cp.fw.featureflags.test.AssumeFlagEnabled; import com.manh.cp.fw.query.Query; import com.manh.cp.schedule.calendar.api.service.CalendarDefinitionService; import com.manh.cp.schedule.utilization.fetch.FetcherFactory; @@ -118,6 +120,9 @@ public class AppointmentDaylightChangeParameterizedTest extends ParameterizedTes context.setTransactionAttribute("component-test", FeatureIdentifier.FeatureFlags24_3.CONSIDER_DAY_LIGHT_SAVINGS_FOR_SOUTH_AMERICA.getFeatureId(), true); context.setTransactionAttribute("component-test", FeatureFlagConstants.FF_99_9_FIX_FACILITY_DAY_LEVEL_SCOPE_FOR_NON_UTC_TIMEZONE, true); dataRemover.removeAllData(context); clearCache("CalendarDefinition:DomainEntity:retail"); </pre>	<p>why do we need to set this ? resolve sonar issues please we may be removing 'CONSIDER_DAY_LIGHT_SAVING_CHANGE' FF. 'CONSIDER_DAY_LIGHT_SAVINGS_FOR_SOUTH_AMERICA' will be used to FF protect the daylight saving changes. move to new FF format we are moving to. with this we will have some data in 1972 date and some in current year. we have tested with mix of data , only old data format and only new data format? move ff to penalty box. we can enabled for customer if they face this issue. We have tested with new data format. This will be called only when a client is creating/editing the Facility Day Level Scope.</p>
4acc32ac9bf2dde41083bac73dc15f89cc69345d	<pre> diff --git a/build.gradle b/build.gradle index 71e710c22..4ef6125d0 100644 --- a/build.gradle +++ b/build.gradle @@ -64,8 +64,8 @@ project.ext.set("gitRevision", org.ajoberstar.grgit.Grgit.open(dir).project.root dependencies { implementation project(":appointment-client") - annotationProcessor(libraries.javax_annotation_api) - annotationProcessor(libraries.javax_persistence) + annotationProcessor(libraries.jakarta_annotation_api) + annotationProcessor(libraries.jakarta_persistence) annotationProcessor(libraries.querydslapt) annotationProcessor(libraries.querydslapt_jpa) diff --git a/client/appointment-client.gradle b/client/appointment-client.gradle index ae91b4c6d..54634be42 100644 --- a/client/appointment-client.gradle +++ b/client/appointment-client.gradle @@ -41,7 +41,7 @@ dependencies { implementation(libraries.jackson_core) implementation(libraries.jackson_databind) implementation(libraries.jackson_datatype_jsr310) - implementation(libraries.javax_annotation_api) + implementation(libraries.jakarta_annotation_api) implementation(libraries.fw_entity_client) implementation(libraries.fw_codegen_model) diff --git a/gradle.properties b/gradle.properties index b136b3610..418f1cea5 100644 --- a/gradle.properties </pre>	<p></p> <p></p> <p></p>

97ac5a506b358d29c298aecadd41060d62cecc4b	<pre> diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentTimeSlotTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentTimeSlotTest.java index 461e6bd19..8552ac9d1 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentTimeSlotTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentTimeSlotTest.java @@@ -211,6 +211,50 @@@ public class AppointmentTimeSlotTest extends AbstractComponentTest "TimeIntervalDefinition_IPOATWEPTS").size()); } + /** + * Test to validate total number of time slots created given time interval definition configured as 00:00 - 10:00 & + * 10:00 - 15:00 & 15:00 - 24:00 with interval of 60 for all days ----> 7 * (10 * (60 / 60)) = 70 ----> 7 * (5 * (60 / + * 60)) = 35 ----> 7 * (9 * (60 / 60)) = 63 Total : = 168 + */ + @Test + public void testPopulationOfAppointmentTimeSlotWithAllDaysUsingMultipleTimeIntervalDef() + { + Document timeIntervalDefinitionDocument = JsonUtil.readDocument(+ "appointment/appointment/appointmentTimeSlot/testPopulationOfAppointmentTimeSlotWithAllDaysAndMoreTimeIntervalDef/TimeIntervalDefinition.json" +); + timeIntervalDefinitionMapService.saveTimeIntervalDefinition(context, timeIntervalDefinitionDocument, null); + Document timeIntervalDefinitionDocument1 = JsonUtil.readDocument(+ "appointment/appointment/appointmentTimeSlot/testPopulationOfAppointmentTimeSlotWithAllDaysAndMoreTimeIntervalDef/TimeIntervalDefinition1.json" +); + timeIntervalDefinitionMapService.saveTimeIntervalDefinition(context, timeIntervalDefinitionDocument1, null); + Document timeIntervalDefinitionDocument2 = JsonUtil.readDocument(</pre>	
c1fce1636d4e6a240fbaf6ffa6c30058c08e69e	<pre> diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/EditAppointmentTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/EditAppointmentTest.java index 14bf5acea..d8983870c 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/EditAppointmentTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/EditAppointmentTest.java @@@ -20,6 +20,8 @@@ package com.manh.cp.appointment.appointment; import java.time.DayOfWeek; import java.time.LocalDateTime; import java.time.LocalTime; +import java.time.ZoneOffset; +import java.time.ZonedDateTime; import java.time.format.DateTimeFormatter; import java.util.ArrayList; import java.util.Arrays; @@@ -57,6 +59,7 @@@ import com.manh.cp.appointment.appointment.client.dto AppointmentDTO; import com.manh.cp.appointment.appointment.client.dto recommendation.RecommendationsOutputDTO; import com.manh.cp.appointment.appointment.constants.AppointmentConstants; import com.manh.cp.appointment.appointment.constants.AppointmentStatus; +import com.manh.cp.appointment.appointment.constants.FeatureFlagConstants; import com.manh.cp.appointment.appointment.constants.FeatureIdentifier; import com.manh.cp.appointment.appointment.impl.service.AppointmentManager; import com.manh.cp.appointment.appointment.impl.service.EntityToDTOConverter; @@@ -83,6 +86,7 @@@ import com.manh.cp.fw.entity.domain.entity.Page; import com.manh.cp.fw.errorHandler.ErrorHandlerService; import com.manh.cp.fw.errorHandler.api.service.ErrorDefinitionService; import com.manh.cp.fw.featureFlags.Features; +import com.manh.cp.fw.featureFlags.test.AssumeFlagEnabled; import com.manh.cp.fw.query.Query; import com.manh.cp.schedule.calendar.api.domain.CalendarScheduleUtilization; import com.manh.cp.schedule.calendar.impl.service.CalendarScheduleUtilizationServiceImpl; @@@ -96,6 +100,7 @@@ import com.manh.cp.schedule.domain.range.DateTimeRange; </pre>	<p>This change is made to correct the test case to demonstrate availability for 0 to 24 hours, as it intermittently fails due to unavailable time slots in the test case, where the time interval definition was initially set to 8-15 hours only.</p> <p>Removed the static 'LocalDateTme' and used dynamic values instead, as we are already passing them while calling the 'scheduleAppointment' method.</p> <p>CI Report:</p> <p></p> <p></p> <p>why do we need this mocking? use AssumeFF</p> <p>@(6152e14e289a54006ab32626) - we cannot rely on the '@AssumeFlagEnabled(FeatureFlagConstants.FF_24_4_FIX_EDIT_APPOINTMENT_DATE)' annotation as it is not working as expected. This may be due to the feature flag service being mocked and set to primary, which will be removed and addressed as part of the full feature flag adoption, as confirmed by @(5f91d9370ce27c008e08e6ac). Therefore, we need to manually set the feature flag for now.</p> <p>@(6152e14e289a54006ab32626) @(557058:4f2d06b7-2446-41af-ab1a-e0d04dee6c9c) We had a requirement where it was asked why are we changing the Schedule Time when we are just updating the Appointment Content. With this change, it will always try to find the Time Slot based on Preferred Date. change it to</p> <pre> !editedAppointmentInputDTO.getPreferredDateTime() isEqual(existingAppointment.getPreferredTime()) </pre> <p>Done.</p>

3cedef3e81ac4104fe3f0a028aafad496ed6b42	<pre> diff --git a/client/src/main/java/com/manh/cp/appointment/appointment/client/Constants/ErrorDefinitionIdBundle.java b/client/src/main/java/com/manh/cp/appointment/appointment/client/Constants/ErrorDefinitionIdBundle.java index fa9b1ccbf..3af64afda 100644 --- a/client/src/main/java/com/manh/cp/appointment/appointment/client/Constants/ErrorDefinitionIdBundle.java +++ b/client/src/main/java/com/manh/cp/appointment/appointment/client/Constants/ErrorDefinitionIdBundle.java @@@ -185,11 @@@ public enum ErrorDefinitionIdBundle OVERLAPPING_PREFERENCE("APT::276"), + APPOINTMENT_TIME_CHANGE_DISABLED_ON_EDIT("APT::277"); + APPOINTMENT_TIME_CHANGE_DISABLED_ON_EDIT("APT::277"); + + REQUIRED_FIELD_MISSING_WITH_APPOINTMENT_ID("APT::278"); + + SCHEDULE_FRAMEWORK_ERROR_WITH_APPOINTMENT_ID("APT::279"); private final String code; diff --git a/gradle.properties b/gradle.properties index 66317d62f..b136b3610 100644 --- a/gradle.properties +++ b/gradle.properties @@@ -1 +1 @@@ -dbChecksum=d97765f28d8f4278f664e65d18964351 +dbChecksum=eb971f93b19c08d1dfa63251627e8d203 diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/CreateTest.java b/src/component- test/java/com/manh/cp/appointment/appointment/CreateTest.java index 52a42cd18..8bed5c4d9 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/CreateTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/CreateTest.java @@@ -50,6 +50,7 @@@ import com.manh.cp.appointment.appointment.client.Constants.ErrorDefinitionIdBun </pre>	<p>Replace 0 and 1 with the name of the item being inserted here avoid FF mock. With minimal FF-test adoption we can use AssumeFF annotation. add fail(!) in case API does not return exception do we need this? avoid FF mock (and other other occurrence in TC add fail(!) in case API does not return exception (and for other api invocation in TC) Add desc which all scenarios we are verifying</p>
0fa74824f9fefec08acfe117866dc587563730d5	<pre> diff --git a/gradle.properties b/gradle.properties index 5092163a0..66317d62f 100644 --- a/gradle.properties +++ b/gradle.properties @@@ -1 +1 @@@ -dbChecksum=158dbf4434a8060b2244b7a11aa8c4e9 +dbChecksum=d97765f28d8f4278f664e65d18964351 diff --git a/src/main/resources/seedData/inBoundMessageType_157__appointment.json b/src/main/resources/seedData/inBoundMessageType_157__appointment.json index 886a3cb3e..4831e0bfe 100644 --- a/src/main/resources/seedData/inBoundMessageType_157__appointment.json +++ b/src/main/resources/seedData/inBoundMessageType_157__appointment.json @@@ -259,7 +259,7 @@@ "NoOfConsumer": 10, "MinNoOfConsumer": 1, "ServiceId": "UpdateResourceGroupConfigurations", "Tags": "dbHighCPU,replicaLagCreator" + "Tags": "dbHighCPU" }, { "MessageType": "YRD_APT_OffYardAppointment", diff --git a/version.txt b/version.txt index 8b0beab16..2b9cab07 100644 --- a/version.txt +++ b/version.txt @@@ -1 +1 @@@ -20.11.0 +20.12.0 </pre>	<p>Lets not do a migration script. Just clean it up from seed. I'll request the DBAs to handle this manually for existing customers. Done.</p>

26d2d44b12c28580549b6ade52bb6a494cfe521	<pre>diff --git a/src/component-test/java/com/manh/cp/appointment/common/ParameterizedTestBase.java b/src/component-test/java/com/manh/cp/appointment/common/ParameterizedTestBase.java index 53839bdbb..f1a4dc554 100644 --- a/src/component-test/java/com/manh/cp/appointment/common/ParameterizedTestBase.java +++ b/src/component-test/java/com/manh/cp/appointment/common/ParameterizedTestBase.java @@ -46,12 +46,14 @@ import com.manh.cp.appointment.core.api.domain.DayLevelUtilization; import com.manh.cp.appointment.core.api.domain.FacilityDayLevelScope; import com.manh.cp.appointment.core.api.domain.FacilityLevelUtilization; import com.manh.cp.appointment.core.api.domain.ResourceGroup; +import com.manh.cp.appointment.core.api.domain.Schedule; import com.manh.cp.appointment.core.api.domain.TimeIntervalDefinition; import com.manh.cp.appointment.core.api.domain.TimeIntervalUtilization; import com.manh.cp.appointment.core.rest.DayLevelUtilizationController; import com.manh.cp.appointment.core.rest.FacilityDayLevelScopeController; import com.manh.cp.appointment.core.rest.FacilityLevelUtilizationController; import com.manh.cp.appointment.core.rest.ResourceGroupController; +import com.manh.cp.appointment.core.rest.ScheduleController; import com.manh.cp.appointment.core.rest.TimeIntervalDefinitionController; import com.manh.cp.appointment.core.rest.TimeIntervalUtilizationController; import com.manh.cp.config.JacksonObjectMapperConfig; @@ -105,6 +107,9 @@ public abstract class ParameterizedTestBase extends AbstractComponentTest @Autowired private FacilityDayLevelScopeController facilityDayLevelScopeController; + @Autowired + private ScheduleController scheduleController; + public Context context; public String testName; @@ -120,6 +125,7 @@ public abstract class ParameterizedTestBase extends AbstractComponentTest</pre>	<p>Do I also need to create a record in appointmentSchedules here?</p> <p>when we define capacity, if no capacity is defined for 'LIVE_UNLOAD', what is our default for total capacity and remaining capacity?</p> <p>did we handle use case where time slots are defined for 30 mins but appointment requested is for 60 min or 90 min? or this use case is still pending to be handled?</p> <p>The time interval capacity must have a capacity defined, but it doesn't need a LIVE_UNLOAD capacity defined. Therefore, we will always have some remaining capacity that we can validate against. I'm assuming that if there isn't a LIVE_UNLOAD capacity defined, the only limit on those appointment types is the remaining capacity, aka, there is no specific limit for that appointment type.</p> <p>That scenario has been accounted for an several test cases already exist for it should we read the appointment id from savedAppointmentDocument instead as its after save and will almost guarantee appointment id being there</p> <p>rename to resourcesWithCapacity</p> <p>If the appointment has skipAvailability check set to false, it might not have capacity. I think approvedResources makes more sense with that field (which has yet to be implemented in this flow)</p> <p>The field was there anyway, but I'll combine 'savedAppointmentDocument' and 'allAppointmentDataDocument' since they should already be the same object and there's no need to have them both</p>
8a493a8f89f9c9d108922bd83af39f776ee6c76b	<pre>diff --git a/gradle.properties b/gradle.properties index 319096b1c..5092163a0 100644 --- a/gradle.properties +++ b/gradle.properties @@ -1,1 +1 @@ -dbChecksum=121d5164d9283e5edb68508ef6794e62 +dbChecksum=158dbf4434a8060b2244b7a11aa8c4e9 diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentCarrierVendorParameterCriteriaTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentCarrierVendorParameterCriteriaTest.java index bd081bbf1..7830cd4cc 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentCarrierVendorParameterCriteriaTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/AppointmentCarrierVendorParameterCriteriaTest.java @@ -22,6 +22,7 @@ import java.time.LocalDateTime; import java.util.ArrayList; import java.util.List; import java.util.Map; +import java.util.Objects; import org.junit.After; import org.junit.Assert; @@ -36,6 +37,7 @@ import org.springframework.transaction.annotation.Transactional; import com.manh.cp.appointment.AbstractComponentTest; import com.manh.cp.appointment.appointment.api.domain.Appointment; import com.manh.cp.appointment.appointment.api.service.AppointmentService; +import com.manh.cp.appointment.appointment.constants.FeatureFlagConstants; import com.manh.cp.appointment.appointment.constants.FeatureIdentifier; import com.manh.cp.appointment.config.api.domain.AppointmentParameterCriteria; import com.manh.cp.appointment.config.api.service.AppointmentParameterCriteriaService; @@ -46,6 +48,7 @@ import com.manh.cp.config.standard.config.api.domain.ConfigRuleGroupXref; import com.manh.cp.config.standard.config.api.service.ConfigRuleGroupXrefService; import com.manh.cp.fw.common.Context;</pre>	<p>can we add Assert.fail() here?</p> <p>No FF mock. Use AssumeFF annotation</p> <p>ensure this is unique across all TC</p> <p>add a message in "fail"</p> <p>reset this to false after TC is done.</p> <p>Add a TC where application param is set to false.</p> <p>can this be added through test seed?</p> <p>can this be done through test seed to avoid this additional code?</p> <p>any where else in hierarchy overrides this in base? as this is the lowest don't see much a concern.</p> <p>Please do check once.</p> <p>please do add initialization check for source type</p> <p>This can not be done through seed data because this can create issue with other test case.</p>

c8ece09049aab49a3563e0270356ff252cab07ec	<pre>diff --git a/gradle.properties b/gradle.properties index c1cfa6731..63e41475e 100644 --- a/gradle.properties +++ b/gradle.properties @@@ -1 +1 @@@ -dbtChecksum=ac7f3b26c811a47ae69ef709965c6ab2 +dbtChecksum=5c161ada0675e25d78b471e00b5194e diff --git a/src/component-test/java/com/manh/cp/appointment/ParameterizedTestBase.java b/src/component-test/java/com/manh/cp/appointment/ParameterizedTestBase.java deleted file mode 100644 index 4891eba96..000000000 --- a/src/component-test/java/com/manh/cp/appointment/ParameterizedTestBase.java +++ /dev/null @@@ -1,45 +0,0 @@@ -/* - * Copyright &#169; 2024 Manhattan Associates, Inc. All Rights Reserved. - * - * Confidential, Proprietary and Trade Secrets Notice - * - * Use of this software is governed by a license agreement. This software - * contains confidential, proprietary and trade secret information of - * Manhattan Associates, Inc. and is protected under United States and - * international copyright and other intellectual property laws. Use, disclosure, - * reproduction, modification, distribution, or storage in a retrieval system in - * any form or by any means is prohibited without the prior express written - * permission of Manhattan Associates, Inc. - * - * Manhattan Associates, Inc. - * 2300 Windy Ridge Parkway, 10th Floor - * Atlanta, GA 30339 USA - */</pre>	<p>we can remove "apt" and "scope" from name of entities. "TimeIntervalUtilisation" Similarly for other entities.</p> <p>Table name 'APT_TIME_INTERVAL_UTILIZATION' add column and table desc to check - Do we need to populate appointment ID? in old fwk it was required, in new it may not be. To check - If we can make parent shipment creation via queue.</p>
d44fe09aad42bdae71ec01ad46ea9373838cc79	<pre>diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/CarrierAppointmentTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/CarrierAppointmentTest.java index 58d518e64..8e2ae4e17 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/CarrierAppointmentTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/CarrierAppointmentTest.java @@@ -45,6 +45,7 @@@ import com.manh.cp.appointment.appointment.client.Constants.ErrorDefinitionIdBun import com.manh.cp.appointment.appointment.constants.FeatureIdentifier; import com.manh.cp.appointment.appointment.impl.service.AppointmentManager; import com.manh.cp.appointment.appointment.impl.service.AppointmentManagerHelper; +import com.manh.cp.appointment.appointment.impl.service.AppointmentSaveHelper; import com.manh.cp.appointment.appointment.mocks.MockAppointmentUtil; import com.manh.cp.appointment.appointment.mocks.MockFeatureFlagService; import com.manh.cp.appointment.appointment.rest.AppointmentController; @@@ -145,6 +146,9 @@@ public class CarrierAppointmentTest extends AbstractComponentTest @Autowired private ApplicationParameterService applicationParameterService; + @Autowired + private AppointmentSaveHelper appointmentSaveHelper; + private final String testDirectory = "/appointment/appointment/carrierAppointment"; @Before @@@ -662,4 +666,58 @@@ public class CarrierAppointmentTest extends AbstractComponentTest Boolean.FALSE.toString()); } + @Test + @Transactional + public void testCarrierScheduleAppointmentWithBulkPurchaseOrders() + {</pre>	<p>pls avoid the * imports can we have a more specific name for the app param? 2024? The flush can also be forced by</p> <p>`manhEntityManagerProvider.getEntityManager().flush()' we need to keep the original method accessible while overriding. One way is to set context transactional attribute from the flow logic and if absent revert to the super class method. with this sort of delete the generated method is still fetching by BK and then deleting one by one, so we'll pay the price of pre-query flush multiple times. Explore JPADelete with In clause log doesnt match match name wont this method be called lots of times. suggest changing to debug level. avoid * import appointment.appointmentContent.saveWithoutLoad /SaveOptimisation to keep it generic for all appointment contents</p>

4ea394f68edacf25c6b41658737b12cfefda78a1	<pre>diff --git a/src/main/java/com/manh/cp/appointment/core/impl/service/ResourceAvailabilityServiceImpl.java b/src/main/java/com/manh/cp/appointment/core/impl/service/ResourceAvailabilityServiceImpl.java index 09c5502ba..185108477 100644 --- a/src/main/java/com/manh/cp/appointment/core/impl/service/ResourceAvailabilityServiceImpl.java +++ b/src/main/java/com/manh/cp/appointment/core/impl/service/ResourceAvailabilityServiceImpl.java @@@ -1171,10 +1171,14 @@@ public class ResourceAvailabilityServiceImpl<T extends AppointmentInputDTO> Impl } // Ensure windowStartDateTime is not in the past relative to the current timestamp - if ((featureFlagService.isOn(FeatureIdentifier.FeatureFlags24_3.AVOID_RECOMMENDING_PAST_DATE_TIMESLOTS) + && windowStartDateTime.isBefore(LocalDateTime.now(ZoneOffset.UTC))) + if ((featureFlagService.isOn(FeatureIdentifier.FeatureFlags24_3.AVOID_RECOMMENDING_PAST_DATE_TIMESLOTS)) { windowStartDateTime = LocalDateTime.now(ZoneOffset.UTC); + // Convert current LocalDateTime to UTC TimeZone + LocalDateTime currentLocalDateTime = ZonedDateTime.now(ZoneOffset.UTC).toLocalDateTime(); + if (windowStartDateTime.isBefore(currentLocalDateTime)) { windowStartDateTime = currentLocalDateTime; } } if (appointmentInputDTO.getWindowStartDateTime() != null &&</pre>	
aab97ad907a4f75829db8a0c13c1100dbb2cfdde	<pre>diff --git a/build.gradle b/build.gradle index 8b39d38de..71e710c22 100644 --- a/build.gradle +++ b/build.gradle @@@ -224,6 +224,12 @@@ tasks.register("runCodeGenerator", JavaExec) { args '-gtd' args 'true' + args '-gh' + args 'false' + args '-gtcc' + args 'true' + args '-lev' + args 'true'</pre>	<p>What is the use case of 'args -gtcc'? Except dcinventory, no other components having this entry, for now I have added all which were used in SB1 version. For other components also , you may need to check in SB1 branch if it was used and missed in SB2</p> <p>We need to add gtcc as well. It was missed for other components during the SB2 move. I will be rolling out an adoption for that in a bit. gh/gfb is a lot more critical.</p>
8f0a90296125be546416d56ccc90faa7eaa4e20	<pre>diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/RecommendationTests.java b/src/component-test/java/com/manh/cp/appointment/appointment/RecommendationTests.java index 6cd9031bf..f778a7932 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/RecommendationTests.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/RecommendationTests.java @@@ -54,6 +54,7 @@@ import com.manh.cp.appointment.appointment.client.Constants.ErrorDefinitionIdBun import com.manh.cp.appointment.appointment.client.dto.recommendation.RecommendationDTO; import com.manh.cp.appointment.appointment.client.dto.recommendation.RecommendationsOutputDTO; import com.manh.cp.appointment.appointment.constants.CapacityConstraintLevel; +import com.manh.cp.appointment.appointment.constants.FeatureFlagConstants; import com.manh.cp.appointment.appointment.constants.FeatureIdentifier; import com.manh.cp.appointment.appointment.impl.service.AppointmentManager; import com.manh.cp.appointment.appointment.impl.service.AppointmentManagerHelper; @@@ -79,6 +80,7 @@@ import com.manh.cp.fw.errorHandler.ErrorHandlerService; import com.manh.cp.fw.errorHandler.api.service.ErrorDefinitionService; import com.manh.cp.fw.errorHandler.exception.ErrorHandlerRuntimeException; import com.manh.cp.fw.featureFlags.Features; +import com.manh.cp.fw.featureFlags.test.AssumeFlagEnabled; import com.manh.cp.fw.query.Query; import com.manh.cp.fw.scoring.config.api.service.PreferenceMapService; import com.manh.cp.schedule.calendar.api.service.CalendarDefinitionMapService; @@@ -202,6 +204,9 @@@ public class RecommendationTests extends AbstractComponentTest context.setTransactionAttribute("component-test", FeatureIdentifier.FeatureFlags99_9.CONSIDER_DAY_LIGHT_SAVING_CHANGE.getFeatureId(), true); + context.setTransactionAttribute("component-test", + FeatureIdentifier.FeatureFlags24_3.AVOID_RECOMMENDING_PAST_DATE_TIMESLOTS.getFeatureId(), + true); } @After</pre>	<p>no manual setting of FF</p> <p>Feature flag adoption PR for the appointment component has not yet been merged. should we make it equal to requested date time if that's the case? @{{6152e14e289a54006ab32626}} @{{5bedeffcb7d54117988cc848}} @{{5ea829cdc5c6230baaf187f3}} - As discussed with @{{5bedeffcb7d54117988cc848}} , we should display the latest time slots relative to the current time if EarliestStartDateTime is before the current timestamp; otherwise, EarliestStartDateTime won't be useful.</p> <p>CC: @{{6152e14e289a54006ab32626}}</p> <p>CI Reports:</p> <p></p> <p></p> <p>@{{631ae2c862fe1e6eac6e6274}} - minimal should have been merged already. You can follow new format for FF in TC</p> <p>@{{6152e14e289a54006ab32626}} - we cannot rely on the '@AssumeFlagEnabled(FeatureFlagConstants.AVOID_RECOMMENDING_PAST_DATE_TIMESLOTS)' annotation as it is not working as expected. This may be due to the feature flag service being mocked and set to primary, which will be removed and addressed as part of the full feature flag adoption, as confirmed by @{{5f91d9370ce27c006e08e6ac}} . Therefore, we need to manually set the feature flag for now.</p> <p>This change was made because the test case intermittently fails. All the test cases within 'ScheduleCarrierAppointmentTest' use the same configurations, with some test cases sharing the same appointment schedule. Running the test cases in parallel may block the time slots, leading to failures in recommendation assertions.</p> <p>CC: @{{6152e14e289a54006ab32626}} , @{{616e5fdf3e3753006f85c477}}</p> <p>Fetch LocalDateTime.now() for UTC and check validation against that.</p> <p>Done!</p>

1726ba0c9980202ff423428aa46c579d9ee36e03	<pre> diff --git a/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentServiceImpl.java b/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentServiceImpl.java index fd49cc69d..66dfff4fc 100644 --- a/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentServiceImpl.java +++ b/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentServiceImpl.java @@@ -17,6 +17,7 @@@ + +package com.manh.cp.appointment.appointment.impl.service; + +import java.time.Duration; +import java.time.LocalDateTime; +import java.time.LocalDate; +import java.util.ArrayList; + +@@@ -818,6 +819,7 @@@ public class AppointmentServiceImpl extends AppointmentServiceImplBase implement +QCalendarScheduleEntity qCalendarScheduleEntity = QCalendarScheduleEntity.calendarScheduleEntity; + + Map<String, CalendarSchedule> allSchedules; + LocalDateTime currentTimeBeforeGettingCalSchedules = LocalDateTime.now(); + + if (featureFlagService.isOn(FeatureIdentifier.FeatureFlags24_2.REMOVE_LIST_RESULTS_USAGE)) + { + @@@ -841,6 +843,10 @@@ public class AppointmentServiceImpl extends AppointmentServiceImplBase implement + (existing, replacement) -> existing); + } + + + logger.debug(+ + " getAllUtilizationsForEachDay() time taken to get fw schedule calendarScheduleEntity from db = {}", + + Duration.between(currentTimeBeforeGettingCalSchedules, LocalDateTime.now()).toMillis()); + + List<String> allScheduleIds = new ArrayList<>(); + allSchedules </pre>	
259e4d6596ca63e2068bbe51e4671d706bb9472a	<pre> diff --git a/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentManagerHelper.java b/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentManagerHelper.java index 1b9225d93..8dc75bfc4 100644 --- a/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentManagerHelper.java +++ b/src/main/java/com/manh/cp/appointment/appointment/impl/service/AppointmentManagerHelper.java @@@ -941,12 +941,13 @@@ public class AppointmentManagerHelper + { + appointmentDTO.getAsn().forEach(asnDTO -> asnDTO.setPk(null)); + } + + boolean isSupportAddlPoFieldsPersistence = featureFlagService + + .isOn(FeatureIdentifier.FeatureFlags24_2.SUPPORT_ADDL_PO_FIELDS_PERSISTENCE); + + if (CollectionUtils.isNotEmpty(appointmentDTO.getPurchaseOrder())) + { + appointmentDTO.getPurchaseOrder().forEach(purchaseOrderDTO -> { + purchaseOrderDTO.setPk(null); + - if (featureFlagService + - .isOn(FeatureIdentifier.FeatureFlags24_2.SUPPORT_ADDL_PO_FIELDS_PERSISTENCE)) + + if (isSupportAddlPoFieldsPersistence) + { + purchaseOrderIdDTOMap + .put(purchaseOrderDTO.getPurchaseOrderId(), purchaseOrderDTO); + @@@ -971,8 +972,7 @@@ public class AppointmentManagerHelper + { + appointmentContentsDTO + .setPurchaseOrder(appointmentContentsDTO.getPurchaseOrderId().getPurchaseOrderId()); + - if (featureFlagService.isOn(+ - FeatureIdentifier.FeatureFlags24_2.SUPPORT_ADDL_PO_FIELDS_PERSISTENCE) + + if (isSupportAddlPoFieldsPersistence + + && MapUtils.isNotEmpty(purchaseOrderIdDTOMap)) + { + PurchaseOrderDTO purchaseOrderDTO = purchaseOrderIdDTOMap </pre>	<p>CI Reports:</p> <p></p> <p></p> <p>why do we need to change TC?</p> <p>@(6152e14e289a54006ab32626) - This change was made because the test case intermittently fails. All the test cases within 'ScheduleCarrierAppointmentTest' use the same configurations, with some test cases sharing the same appointment schedule. Running the test cases in parallel may block the time slots, leading to failures in recommendation assertions.</p> <p>CC: @(616e5fdf3e3753006f85c4f7)</p> <p>we can log the time the process started here, and it's end at the end of this method. We are already getting the execution time as part of the API response because we have annotated the method with '@TrackTime(value = TrackTime.DEBUG, skipArgs = true, skipResult = true)'.</p> <p></p>

<p>eaeee88fd31f620451b7e0d94302793da4c0f041</p>	<pre>diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentInDayLightSavingsTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentInDayLightSavingsTest.java index 2e8a9bdd9..cb76031cd 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentInDayLightSavingsTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/DaylightSavings/AppointmentInDayLightSavingsTest.java @@ -1154,8 +1154,7 @@ public class AppointmentInDayLightSavingsTest extends AbstractComponentTest calendarRequestDocument.put("CalendarDate", calendarRequestTimeUTC.toString()); - + RestApiResponse calendarResponse = + null; + RestApiResponse calendarResponse = null; String appointmentPath = "/appointment/appointment/AppointmentInDayLightSavings/calendarTestOnDayLightSavingEnd/Appointment.json"; diff --git a/src/component-test/java/com/manh/cp/appointment/appointment/InboundDeliveryTest.java b/src/component-test/java/com/manh/cp/appointment/appointment/InboundDeliveryTest.java index 9291a754e..1da064080 100644 --- a/src/component-test/java/com/manh/cp/appointment/appointment/InboundDeliveryTest.java +++ b/src/component-test/java/com/manh/cp/appointment/appointment/InboundDeliveryTest.java @@ -115,7 +115,7 @@ public class InboundDeliveryTest extends AbstractComponentTest createConfiguration(); testUtil.createAppointment2(context, "/appointment/appointment/InboundShipment/InboundShipmentAppointment.json", - LocalDateTime.now().plusDays(1).toLocalDate().atTime(1, 0).toString()); + LocalDateTime.now().plusDays(2).toLocalDate().atTime(1, 0).toString()); Query query = new Query(" ((AppointmentContents.InboundShipment _ 'Inbound%')) ", 0, 10); @@ -161,7 +161,7 @@ public class InboundDeliveryTest extends AbstractComponentTest createConfiguration();</pre>	<p>Same question here, do we want to log the entire 'AppointmentScheduleRecommendationsDTO' ? Also, method should be changed to 'scheduleAppointmentWithNewConfig'()</p> <p>The previous 'scheduleAppointment' method has these annotations:</p> <pre>... @Transactional(propagation = Propagation.REQUIRED) @TrackTime(value = TrackTime.DEBUG, skipArgs = true, skipResult = true) ...</pre> <p>Are these no longer necessary?</p> <p>Do we want to log the entire 'enhancedAppointmentDTO' as well as the initial document that we're passing in? Not sure how memory intensive that would be, but instead maybe there are only a few key fields that we would need to validate from both?</p> <p>I might be misunderstanding the point of the refactoring, but I thought we would want a new DTO that is more pared down or substantively different from the AppointmentDTO right? If we're just extending the same AppointmentDTO that we've been using this whole time, then why not just add more fields to that DTO itself?</p> <p>Change trace to 'getEnhancedAppointmentDTO'()</p> <p>Trace and debug logs should have method changed to 'scheduleAppointmentWithNewConfig'()</p> <p>Were you able to invoke 'scheduleAppointmentUserExit.preprocess' ?</p> <p>No, and I'm not going to fix it in this PR, but it'll get fixed before this branch gets merged to master</p> <p>This is what we're doing in the old flow and for consistency, I'm going to keep it the same for now. After this flow is complete, I'll go through and remove what isn't necessary, but for now, I don't know what will/won't make the cut</p> <p>This is a holdover from a previous version that condensed AppointmentDTO with FacilityAppointmentInputDTO and AppointmentInputDTO. Those changes are now gone, but I'm keeping this here in case there are other fields I want to add to the new flow while it is being built.</p> <p>There's also a decent chance it just gets deleted after all it is complete</p>
<p>af9fa033fc9e5fee1506a0ae7178048359f38d5c</p>	<pre>diff --git a/src/main/java/com/manh/cp/appointment/appointment/constants/AppointmentConstants.java b/src/main/java/com/manh/cp/appointment/appointment/constants/AppointmentConstants.java index 650152c60..a5feb3e16 100644 --- a/src/main/java/com/manh/cp/appointment/appointment/constants/AppointmentConstants.java +++ b/src/main/java/com/manh/cp/appointment/appointment/constants/AppointmentConstants.java @@ -277,4 +277,7 @@ public class AppointmentConstants public static final String REQUESTOR_CONTACT = "RequestorContact"; public static final String FLOW_TYPE = "FlowType"; + + public static final String NOT_ON_BOARDED = "Not-On-Boarded"; + + } diff --git a/src/main/java/com/manh/cp/appointment/appointment/impl/service/CarrierService.java b/src/main/java/com/manh/cp/appointment/appointment/impl/service/CarrierService.java index 7b1075502..78ae6b87 100644 --- a/src/main/java/com/manh/cp/appointment/appointment/impl/service/CarrierService.java +++ b/src/main/java/com/manh/cp/appointment/appointment/impl/service/CarrierService.java @@ -19,6 +19,7 @@ package com.manh.cp.appointment.appointment.impl.service; import java.text.MessageFormat; +import org.apache.commons.lang3.ObjectUtils; +import org.apache.commons.lang3.StringUtils; +import org.springframework.beans.factory.annotation.Autowired; +import org.springframework.http.HttpStatus; @@ -28,6 +28,7 @@ import org.springframework.web.client.HttpClientErrorException; import com.manh.cp.appointment.appointment.ExternalServiceLocator; import com.manh.cp.appointment.appointment.constants.AppointmentConstants; +import com.manh.cp.carrier.carrier.client.dto.CarrierBaseDTO;</pre>	<p>Add test case.</p> <p>This is an existing functionality, and already test cases would've ran for the same (build passed!). We have just added a null check here.</p> <p>@(618ad60c10bdeb007067125b) : Don't we need isInitialised check since this is a API call. Have we tested ?</p> <p>@(5fbf6d2facfd60076fd072b) : Ive added the isInitialised() check, but the issue here was caused by NPE (logs attached below), as the 'OnBoardingStatusId' key's value was coming as null, and it failed at :</p> <p>'carrierDTO.getOnBoardingStatusId().getOnBoardingStatusId()' since 'carrierDTO.getOnBoardingStatusId()' was null.</p> <p>Testing done:</p> <p>When the OnboardingStatusId was null → NPE</p> <p>When the OnboardingStatusId was On-Boarded or Not-On-Boarded → Flow worked.</p> <p>_Handled by CustomErrorController java.lang.NullPointerException: null at com.manh.cp.appointment.appointment.impl.service.CarrierService.isCarrierOnboarded(CarrierService.java:84) ~[classes/:na] at com.manh.cp.appointment.appointment.publisher.AppointmentContentsPublisher.publishAppointmentUpdateToCarrierPortal(AppointmentContentsPublisher.java:252) ~[classes/:na]_</p> <p>Add TC to cover these scenarios</p> <p>@(6152e14e289a54006ab32626) - Since my changes are only for null check for onboarding status id in carrierDTO, I don't think a dedicated TC is required to verify the null check. Pls let me know if any concern.</p> <p>Cc : @(5f8cb08a6bc6340068b81fa1) @(5ac30fd67b9e19039583e0f6)</p> <p>@(618ad60c10bdeb007067125b) - please add TC to verify flow is working as expected for different onboarding status.</p>

7c4a8faa6431b0390253e9dbaaba58d401647fd9	<pre> diff --git a/build.gradle b/build.gradle index b56604a6c..8b39d38de 100644 --- a/build.gradle +++ b/build.gradle @@@ -169,6 +169,7 @@@ dependencies { } testImplementation(libraries.fw_entity_test) + testImplementation(libraries.fw_feature) testImplementation(libraries.fw_test) testImplementation(libraries.fw_dm_test_common) testImplementation(libraries.h2_database) diff --git a/libraries.gradle b/libraries.gradle index 82993c048..78013e253 100644 --- a/libraries.gradle +++ b/libraries.gradle @@@ -72,6 +72,7 @@@ ext { fw_entity_starter: "com.manh.cp.fw:fw-entity-starter", fw_error_handler: "com.manh.cp.fw:fw-error-handler", fw_feature_flags: "com.manh.cp.fw:fw-feature-flags", + fw_feature: "com.manh.cp.fw:fw-feature-flags-test", fw_grant: "com.manh.cp.fw:fw-grant", fw_localization: "com.manh.cp.fw:fw-localization", fw_log_admin: "com.manh.cp.fw:fw-log-admin", diff --git a/version.txt b/version.txt index f3f52b42d..d5a159609 100644 --- a/version.txt +++ b/version.txt @@@ -1 +1 @@@ -20.9.0 +20.10.0 </pre>	
--	--	--