BQS Patient Data Service - Full API Documentation (Story IHS3-468)

🔴 Old Endpoint: /getPrefillData [POST]

Description: Legacy API used for department-specific prefill data retrieval.

Sample Request Body:

{  
 "applicationId": "12345",  
 "ssn": "123-45-6789"  
}

Controller Snippet:

@PostMapping("/getPrefillData")  
public ResponseEntity<PrefillResponse> getData(@RequestBody PrefillRequest request) {  
 return ResponseEntity.ok(service.fetchData(request));  
}

🔄 Architectural Shift: Single Proxy ➡ Multiple Proxies

Previously, the system used a single proxy to route all API requests, leading to hardcoded logic, limited flexibility, and tight coupling with departments.

Now, in the enhanced architecture (IHS3-468), we have introduced multiple proxies and dynamic routing for multiple departments.

🚀 Key Improvements in Architecture

- Header-driven proxy routing using `X-SYF-API-ID`

- Modular proxy logic for each department (e.g., Dental, Auto, Hospital)

- New proxy structure supports easy onboarding of new departments

- Reduces logic duplication and improves maintenance

📌 Before vs After: Proxy Usage

🔴 Old (Single Proxy):

@PostMapping("/getPrefillData")  
public ResponseEntity<PrefillResponse> getPrefill(@RequestBody PrefillRequest request) {  
 if ("DENTAL".equals(request.getType())) {  
 return dentalService.getPrefillData(request);  
 } else if ("AUTO".equals(request.getType())) {  
 return autoService.getPrefillData(request);  
 }  
 return defaultService.getPrefillData(request);  
}

🟢 New (Multiple Proxies + Department Routing):

@PostMapping("/bqs-prefill-data")  
public ResponseEntity<GenericResponse<PrefillResponse>> getPrefillData(  
 @RequestBody @Valid PrefillRequest request,  
 @RequestHeader("X-SYF-API-ID") String apiId) {  
  
 switch (apiId) {  
 case "AUTO": return autoProxy.getPrefillData(request);  
 case "DENTAL": return dentalProxy.getPrefillData(request);  
 case "HOSPITAL": return hospitalProxy.getPrefillData(request);  
 default: throw new IllegalArgumentException("Unsupported API ID");  
 }  
}

🧠 Use of Generic Structure for All Endpoints

- GenericResponse<T> used for all service responses

- Unified error handling with `@RestControllerAdvice`

- Swagger-compatible annotations with detailed error codes

✅ Updated Endpoint Flow: All Departments & Proxies

- `/bqs-prefill-data` → replaces all old department-specific endpoints

- Logic based on API ID header → directs to correct department proxy

- New services follow open/closed principle for adding more verticals

🟢 New Equivalent: /bqs-prefill-data

🔹 Request:

{  
 "applicationId": "12345",  
 "state": "CA",  
 "ssn": "123-45-6789"  
}

🔹 Response:

{  
 "status": "SUCCESS",  
 "data": {  
 "patientId": "P12345",  
 "fullName": "John Doe",  
 "dob": "1990-01-01"  
 }  
}

🔹 Mandatory Fields:

- applicationId  
- X-SYF-API-ID  
- X-SYF-API-KEY

🔹 Swagger Annotations:

@Operation(summary = "Generic prefill endpoint")  
@ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Data retrieved"),  
 @ApiResponse(responseCode = "400", description = "Bad Request"),  
 @ApiResponse(responseCode = "500", description = "Internal Server Error")  
})

🔹 Error Response Example:

{  
 "status": "FAILURE",  
 "errorCode": "400\_BAD\_REQUEST",  
 "message": "Missing applicationId"  
}

🟢 New Equivalent: /bqs-prefill-data (X-SYF-API-ID=DENTAL)

🔹 Request:

{  
 "applicationId": "DENTAL\_001",  
 "toothCode": "T12"  
}

🔹 Response:

{  
 "status": "SUCCESS",  
 "data": {  
 "patientName": "Jane Smith",  
 "toothCode": "T12",  
 "treatment": "Cavity Fill"  
 }  
}

🔹 Mandatory Fields:

- applicationId  
- toothCode  
- X-SYF-API-ID = DENTAL

🔹 Swagger Annotations:

@Schema(description = "Dental prefill input")

🔹 Error Response Example:

{  
 "status": "FAILURE",  
 "errorCode": "404\_NOT\_FOUND",  
 "message": "No record found for applicationId"  
}

🟢 New Equivalent: /bqs-prefill-data (X-SYF-API-ID=AUTO)

🔹 Request:

{  
 "applicationId": "AUTO\_001",  
 "licensePlate": "MH12XY1234"  
}

🔹 Response:

{  
 "status": "SUCCESS",  
 "data": {  
 "owner": "Rahul Kumar",  
 "vehicle": "Honda City",  
 "year": 2020  
 }  
}

🔹 Mandatory Fields:

- applicationId  
- licensePlate  
- X-SYF-API-ID = AUTO

🔹 Swagger Annotations:

@Schema(description = "Auto prefill input")

🔹 Error Response Example:

{  
 "status": "FAILURE",  
 "errorCode": "400\_INVALID\_LICENSE",  
 "message": "Invalid license plate"  
}

📘 Swagger Annotations & Request Field Specifications

🧩 Complete Swagger Annotations for Generic Prefill Endpoint

@Tag(name = "Patient Prefill API", description = "Endpoints for fetching prefill data for various departments.")  
  
@Operation(  
 summary = "Fetch Prefill Data",  
 description = "Returns department-wise prefill data based on API header and request body."  
)  
  
@Parameters({  
 @Parameter(name = "X-SYF-API-ID", description = "Department ID e.g., DENTAL, AUTO, HOSPITAL", required = true, in = ParameterIn.HEADER),  
 @Parameter(name = "X-SYF-API-KEY", description = "Authentication Key", required = true, in = ParameterIn.HEADER)  
})  
  
@ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Prefill data retrieved successfully", content = @Content(schema = @Schema(implementation = PrefillResponse.class))),  
 @ApiResponse(responseCode = "400", description = "Invalid request parameters"),  
 @ApiResponse(responseCode = "401", description = "Unauthorized"),  
 @ApiResponse(responseCode = "500", description = "Internal server error")  
})

📝 Prefill Request Parameters with Annotations

@JsonProperty("applicationId") @NotBlank  
private String applicationId;  
  
@JsonProperty("state") @NotBlank  
private String state;  
  
@JsonProperty("ssn") @Pattern(regexp = "\\d{3}-\\d{2}-\\d{4}")  
private String ssn;  
  
@JsonProperty("firstName")  
private String firstName;  
  
@JsonProperty("lastName")  
private String lastName;  
  
@JsonProperty("dob") @Pattern(regexp = "\\d{4}-\\d{2}-\\d{2}")  
private String dob;

🧪 Java Validation & DTO Code Blocks

✅ PrefillRequest.java with Validations

// PrefillRequest.java - Request DTO with validation  
  
public class PrefillRequest {  
  
 @JsonProperty("applicationId")  
 @NotBlank(message = "Application ID is mandatory")  
 private String applicationId;  
  
 @JsonProperty("state")  
 @NotBlank(message = "State is required")  
 private String state;  
  
 @JsonProperty("ssn")  
 @Pattern(regexp = "\d{3}-\d{2}-\d{4}", message = "SSN must follow XXX-XX-XXXX format")  
 private String ssn;  
  
 @JsonProperty("firstName")  
 private String firstName;  
  
 @JsonProperty("lastName")  
 private String lastName;  
  
 @JsonProperty("dob")  
 @Pattern(regexp = "\d{4}-\d{2}-\d{2}", message = "DOB must be in YYYY-MM-DD format")  
 private String dob;  
}

📦 PrefillResponse.java Structure

// PrefillResponse.java - Response DTO  
  
public class PrefillResponse {  
  
 private String patientId;  
 private String fullName;  
 private String dob;  
 private String department;  
}

# ✅ Updated Technical Improvements & Additions

## 📌 Required Request Headers

The new generic API expects the following headers in every request:  
  
- X-SYF-API-ID: Department Identifier (e.g., AUTO, DENTAL, HOSPITAL, etc.)  
- X-SYF-API-KEY: Secure API key (Optional, if enabled)  
- Content-Type: application/json

## 📘 Swagger Annotations (Extended)

Example annotations used in the new generic controller:  
  
@Operation(summary = "Generic prefill endpoint")  
@Parameter(name = "X-SYF-API-ID", description = "Identifies which department's data to fetch", required = true)  
@ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Data returned successfully"),  
 @ApiResponse(responseCode = "400", description = "Invalid input or header")  
})  
@Tag(name = "Patient Prefill API")

## 📑 Mandatory Fields with Validations

Example of annotated model fields with validation:  
  
@JsonProperty("dob")  
@Pattern(regexp = "\\d{4}-\\d{2}-\\d{2}", message = "DOB must be in YYYY-MM-DD format")  
private String dob;  
  
@JsonProperty("applicationId")  
@NotBlank(message = "Application ID is mandatory")  
private String applicationId;

## 🛡️ Global Exception Handling

@RestControllerAdvice  
public class GlobalExceptionHandler {  
  
 @ExceptionHandler(IllegalArgumentException.class)  
 public ResponseEntity<ErrorResponse> handleInvalidHeader(IllegalArgumentException ex) {  
 return ResponseEntity.badRequest().body(new ErrorResponse("400\_BAD\_REQUEST", ex.getMessage()));  
 }  
}

## 🔁 Legacy Controller Logic vs Generic Header Routing

|  |  |
| --- | --- |
| 🔴 Legacy Logic (Controller if-else) | 🟢 Generic Routing (Header-based) |
| if (type.equals("AUTO")) | X-SYF-API-ID=AUTO |
| if (type.equals("DENTAL")) | X-SYF-API-ID=DENTAL |
| if (type.equals("HOSPITAL")) | X-SYF-API-ID=HOSPITAL |

## 📦 Request JSON Schema Explanation

Example schema from Swagger/OpenAPI:  
  
{  
 "applicationId": {  
 "type": "string",  
 "required": true,  
 "description": "Unique application identifier"  
 },  
 "state": {  
 "type": "string",  
 "required": true,  
 "description": "State code (e.g., CA, NY)"  
 },  
 "dob": {  
 "type": "string",  
 "format": "date",  
 "required": false,  
 "description": "Date of birth"  
 }  
}