NTAAT Scheduler - Technical Documentation

1. Executive Overview

The NTAAT Scheduler is a workload optimization tool designed to automatically assign sites to Registered Nurses (RNs) and Provider Teams based on daily patient census and acuity categories.

Architecture:
- Platform: Microsoft Excel (Enterprise Edition with Python enabled).
- Engine: Native Python in Excel (=PY()) running in the Microsoft Cloud.
- Input Method: Standard Excel Tables.
- Output Method: Python DataFrames rendered as Excel Values.

2. Setup & Requirements

The tool relies on two named Excel Tables. These names are hardcoded and must not be changed.

A. Input Table (Name: InputTable)
- Location: 'Inputs' Tab
- Required Rows: 'Num_RNs', 'Num_Providers', 'APRN_Active' (Yes/No)

B. Census Data Table (Name: SiteTable)
- Location: 'Census Input' Tab
- Required Columns: 'SiteCode', 'facilityName', 'TotalCensus', 'Categories'

3. Operational Guide (Daily Workflow)

Step 1: Update Census
Open 'Census Input' tab. Clear old data and paste new census numbers. Ensure no 'Grand Total' row exists.

Step 2: Adjust Staffing
Open 'Inputs' tab. Update 'Num_RNs' or 'Num_Providers'. Toggle 'APRN_Active'.

Step 3: Run Optimization
Updates automatically. If not, press F9 (Calculate Now) or Ctrl+Alt+F9 (Force Reset).

4. Source Code: RN Distribution (Cell A1)

```
import pandas as pd
output_df = pd.DataFrame(["No Data"])
try:
    df_sites = xl("SiteTable[#All]", headers=True)
    df_inputs = xl("InputTable[#All]", headers=True)
    num_rns = int(df_inputs.loc[df_inputs['Variable'] == 'Num_RNs', 'Value'].iloc[0])
    try:
        aprn_val = df_inputs.loc[df_inputs['Variable'] == 'APRN_Active', 'Value'].iloc[0]
        aprn_on = str(aprn_val).lower() in ['yes', '1', 'true', 'on']
    except: aprn_on = False
    aprn_list = ['ASH', 'BEC', 'BHH', 'CAV', 'CHY', 'FAR', 'FAV', 'FHM', 'LEA', 'LEB', 'LKC', 'MAR', 'MUS',
'SUX', 'TOP', 'WIM', 'BIL', 'CLA', 'CTX', 'GRJ', 'MOU', 'WIC', 'BRX', 'CMS', 'DET', 'MWV', 'TOG', 'SBY']
    df_sites = df_sites[df_sites['facilityName'].str.lower() != 'total']
    df_sites['TotalCensus'] = pd.to_numeric(df_sites['TotalCensus'], errors='coerce').fillna(0)
    df_sites = df_sites.sort_values('TotalCensus', ascending=False)
    rns = [{'id': f"RN {i+1}", 'census': 0, 'count': 0} for i in range(num_rns)]
    assignments = []
    for _, row in df_sites.iterrows():
        name = row['facilityName']; cen = row['TotalCensus']; orig = row['SiteCode']
        rns.sort(key=lambda x: x['census'])
        assigned = False
        for rn in rns:
            if (rn['census'] + cen <= 55) and (rn['count'] < 7):
                rn['census'] += cen; rn['count'] += 1
                is_aprn = "Yes" if (aprn_on and orig in aprn_list) else "No"
                assignments.append([rn['id'], name, int(cen), is_aprn])
                assigned = True; break
        if not assigned: assignments.append(["UNASSIGNED", name, int(cen), "No"])
    df_out = pd.DataFrame(assignments, columns=["RN Number", "Site", "Census", "APRN?"])
    output_df = df_out.sort_values(by=['RN Number', 'Census'], ascending=[True, False]).reset_index(drop=True)
except Exception as e: output_df = pd.DataFrame([f"Error: {str(e)}"])
output_df
```

5. Source Code: Provider Teams (Cell A1)

```
import pandas as pd
output_df = pd.DataFrame(["No Data"])
try:
    df_sites = xl("SiteTable[#All]", headers=True)
    df_inputs = xl("InputTable[#All]", headers=True)
    num_teams = int(df_inputs.loc[df_inputs['Variable'] == 'Num_Providers', 'Value'].iloc[0])
    team_4 = {
        1: ['Ann Arbor', 'Asheville', 'Augusta', 'Bay Pines', 'Beckley', 'Bronx', 'Cincinnati', 'Clarksburg',
'Columbia, SC', 'Connecticut', 'Dayton', 'Detroit', 'Huntington','Hampton', 'Lebanon', 'Northern Indiana',
'NYN-New York Harbor','BYN-New York Harbor', 'Salisbury', 'ALN-Upstate New York', 'BUF-Upstate New York',
'SYR-Upstate New York','White River Junction', 'Wilmington', 'Wilkes-Barre'],
        2: ['Albuquerque', 'Amarillo', 'Cheyenne', 'Denver', 'Grand Junction', 'Houston', 'Iowa City',
'Marion', 'Muskogee', 'Nellis AFB DOD', 'Nellis AFB', 'Oklahoma City', 'Palo Alto', 'San Juan', 'Reno',
'Spokane', 'St. Louis', 'Topeka', 'Tucson', 'Wichita'],
        3: ['Atlanta', 'Baltimore', 'Birmingham', 'Central Alabama', 'Fayetteville, NC', 'Indianapolis',
'Martinsburg', 'Memphis', 'Mountain Home', 'North Florida/South Georgia', 'New Jersey', 'Northport', 'Orlando',
'Richmond', 'Togus', 'Washington DC', 'Black Hills', 'Loma Linda'],
        4: ['Biloxi', 'Boise', 'Columbia, MO', 'Des Moines', 'Fargo', 'Fayetteville, AR', 'Fresno', 'Fort
Harrison', 'Jackson', 'Kansas City', 'Las Vegas', 'Leavenworth', 'Little Rock', 'Minneapolis', 'New Orleans',
'Omaha', 'Phoenix', 'Sacramento', 'Shreveport', 'Sioux Falls', 'Temple (Central Texas)', 'West Los Angeles',
'Cleveland']
    }
    rows = []
    if num_teams == 4:
        for team, sites in team_4.items():
            team_data = df_sites[df_sites['facilityName'].isin(sites)].copy()
            if not team_data.empty:
                team_data = team_data.sort_values(by='Categories')
                for _, r in team_data.iterrows():
                    rows.append([f"Team {team}", int(r['Categories']), r['facilityName'],
int(r['TotalCensus'])])
    df_out = pd.DataFrame(rows, columns=["Team", "Cat", "Site", "Census"])
    output_df = df_out.set_index("Team")
except Exception as e: output_df = pd.DataFrame([f"Error: {str(e)}"])
output_df
```

NTAAT Scheduler - Technical Documentation

6. Source Code: Cross-Walk (Cell A1)

```
import pandas as pd
output_df = pd.DataFrame(["No Data"])
try:
    df_sites = xl("SiteTable[#All]", headers=True); df_inputs = xl("InputTable[#All]", headers=True)
    df_sites = df_sites.dropna(subset=['facilityName'])
    df_sites = df_sites[df_sites['facilityName'].str.lower() != 'total']
    df_sites['TotalCensus'] = pd.to_numeric(df_sites['TotalCensus'], errors='coerce').fillna(0)
    df_sites = df_sites.sort_values('TotalCensus', ascending=False)
    num_rns = int(df_inputs.loc[df_inputs['Variable'] == 'Num_RNs', 'Value'].iloc[0])
    num_teams = int(df_inputs.loc[df_inputs['Variable'] == 'Num_Providers', 'Value'].iloc[0])

    rns = [{'id': f"RN {i+1}", 'census': 0, 'count': 0} for i in range(num_rns)]
    rn_map = {}
    for _, row in df_sites.iterrows():
        name = row['facilityName']; cen = row['TotalCensus']
        rns.sort(key=lambda x: x['census'])
        assigned = False
        for rn in rns:
            if (rn['census'] + cen <= 55) and (rn['count'] < 7):
                rn_map[name] = rn['id']; rn['census'] += cen; rn['count'] += 1
                assigned = True; break
        if not assigned: rn_map[name] = "UNASSIGNED"

    prov_map = {}
    team_4 = {
        1: ['Ann Arbor', 'Asheville', 'Augusta', 'Bay Pines', 'Beckley', 'Bronx', 'Cincinnati', 'Clarksburg',
'Columbia, SC', 'Connecticut', 'Dayton', 'Detroit', 'Huntington','Hampton', 'Lebanon', 'Northern Indiana',
'NYN-New York Harbor','BYN-New York Harbor', 'Salisbury', 'ALN-Upstate New York', 'BUF-Upstate New York',
'SYR-Upstate New York','White River Junction', 'Wilmington', 'Wilkes-Barre'],
        2: ['Albuquerque', 'Amarillo', 'Cheyenne', 'Denver', 'Grand Junction', 'Houston', 'Iowa City',
'Marion', 'Muskogee', 'Nellis AFB DOD', 'Nellis AFB', 'Oklahoma City', 'Palo Alto', 'San Juan', 'Reno',
'Spokane', 'St. Louis', 'Topeka', 'Tucson', 'Wichita'],
        3: ['Atlanta', 'Baltimore', 'Birmingham', 'Central Alabama', 'Fayetteville, NC', 'Indianapolis',
'Martinsburg', 'Memphis', 'Mountain Home', 'North Florida/South Georgia', 'New Jersey', 'Northport', 'Orlando',
'Richmond', 'Togus', 'Washington DC', 'Black Hills', 'Loma Linda'],
        4: ['Biloxi', 'Boise', 'Columbia, MO', 'Des Moines', 'Fargo', 'Fayetteville, AR', 'Fresno', 'Fort
Harrison', 'Jackson', 'Kansas City', 'Las Vegas', 'Leavenworth', 'Little Rock', 'Minneapolis', 'New Orleans',
'Omaha', 'Phoenix', 'Sacramento', 'Shreveport', 'Sioux Falls', 'Temple (Central Texas)', 'West Los Angeles',
'Cleveland']
    }
    if num_teams == 4:
        for team, sites in team_4.items():
            for s in sites: prov_map[s] = f"Team {team}"

    crosswalk = []
    for name in sorted(df_sites['facilityName'].unique()):
        rn = rn_map.get(name, "Unassigned")
        prov = prov_map.get(name, "Unassigned")
        crosswalk.append([name, rn, prov])
    df_out = pd.DataFrame(crosswalk, columns=["Site Name", "Assigned RN", "Assigned Provider"])
    output_df = df_out.reset_index(drop=True)
except Exception as e: output_df = pd.DataFrame([f"Error: {str(e)}"])
output_df
```